УНИВЕРЗИТЕТ У БЕОГРАДУ
ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ

Владимир Л. Петровић

ФЛЕКСИБИЛНИ КОДЕР И ДЕКОДЕР КОДОВА СА ПРОВЕРАМА ПАРНОСТИ МАЛЕ ГУСТИНЕ

Докторска дисертација

Београд, 2021.

UNIVERSITY OF BELGRADE
SCHOOL OF ELECTRICAL ENGINEERING

Vladimir L. Petrović

FLEXIBLE ENCODER AND DECODER OF LOW DENSITY PARITY CHECK CODES

Doctoral Dissertation

Belgrade, 2021

Ментор:
др Лазар Сарановац, редовни професор
Универзитет у Београду – Електротехнички факултет

Чланови комисије:
др Јелена Поповић-Божовић, доцент
Универзитет у Београду – Електротехнички факултет

др Горан Ђорђевић, редовни професор
Универзитет у Нишу – Електронски факултет

др Предраг Иваниш, редовни професор
Универзитет у Београду – Електротехнички факултет

др Драгомир Ел Мезени, доцент
Универзитет у Београду – Електротехнички факултет

др Срђан Бркић, доцент
Универзитет у Београду – Електротехнички факултет

Датум одбране:_____

Мојој породици и учитељима Миловану и Срђану

Захвалница

На почетку бих желео да се захвалим колегама са којима сам сарађивао на пројектима из којих су проистекли резултати приказани у докторској дисертацији, а посебно Андреји Радошевићу и Милошу Марковићу из компаније *Tannera* из Лос Анђелеса и Београда и доц. Драгомиру Ел Мезенију, доц. Срђану Бркићу, Ђорђу Сарачу и проф. Предрагу Иванишу са Електротехничког факултета у Београду.

Захваљујем се ментору, проф. Лазару Сарановцу, и члановима комисије на корисним сугестијама у току израде дисертације и у току писања текста, као и другим колегама са Електротехничког факултета у Београду на сарадњи и пруженим саветима.

Највећу захвалност дугујем својој породици, посебно супрузи Јовани, сестри Марији, родитељима Луки и Звездани, теткама Милеси и Боси, као и својим учитељима, Срђану Брадашу и свештенику Миловану Вујовићу, који су током целог мог живота или у кључним тренуцима били уз мене, давали безрезервну подршку и заједно са мном се радовали сваком успеху.

Флексибилни кодер и декодер кодова са проверама парности мале густине

Сажетак

У дисертацији су предложена брза, флексибилна и хардверски ефикасна решења за кодовање и декодовање изузетно нерегуларних кодова са проверама парности мале густине (енгл. *low-density parity-check, LDPC, codes*) захтевана у савременим комуникационим стандардима.

Један део доприноса дисертације је у новој делимично паралелној архитектури LDPC кодера за пету генерацију мобилних комуникација. Архитектура је заснована на флексибилној мрежи за кружни померај која омогућава паралелно процесирање више делова контролне матрице кратких кодова чиме се остварује сличан ниво паралелизма као и при кодовању дугачких кодова. Поред архитектуралног решења, предложена је оптимизација редоследа процесирања контролне матрице заснована на генетичком алгоритму, која омогућава постизање великих протока, малог кашњења и тренутно најбоље ефикасности искоришћења хардверских ресурса.

У другом делу дисертације предложено је ново алгоритамско и архитектурално решење за декодовање структурираних LDPC кодова. Често коришћени приступ у LDPC декодерима је слојевито декодовање, код кога се услед проточне обраде јављају хазарди података који смањују проток. Декодер предложен у дисертацији у конфликтним ситуацијама на погодан начин комбинује слојевито и симултано декодовање чиме се избегавају циклуси паузе изазвани хазардима података. Овај приступ даје могућност за увођење великог броја степени проточне обраде чиме се постиже висока учестаност сигнала такта. Додатно, редослед процесирања контролне матрице је оптимизован коришћењем генетичког алгоритма за побољшане перформансе контроле грешака. Остварени резултати показују да, у поређењу са референтним решењима, предложени декодер остварује значајна побољшања у протоку и најбољу ефикасност за исте перформансе контроле грешака.

Кључне речи: заштитно кодовање, кодови са проверама парности мале густине, пета генерација мобилних комуникација, флексибилност, проточна обрада, архитектура кодера, архитектура декодера, слојевито декодовање, ефикасност искоришћења хардверских ресурса, генетички алгоритам

Научна област: Електротехника и рачунарство

Ужа научна област: Електроника

Flexible Encoder and Decoder of Low-Density Parity-Check Codes

Abstract

The dissertation proposes high speed, flexible and hardware efficient solutions for coding and decoding of highly irregular low-density parity-check (LDPC) codes, required by many modern communication standards.

The first part of the dissertation's contributions is in the novel partially parallel LDPC encoder architecture for 5G. The architecture was built around the flexible shifting network that enables parallel processing of multiple parity check matrix elements for short to medium code lengths, thus providing almost the same level of parallelism as for long code encoding. In addition, the processing schedule was optimized for minimal encoding time using the genetic algorithm. The optimization procedure contributes to achieving high throughputs, low latency, and up to date the best hardware usage efficiency (HUE).

The second part proposes a new algorithmic and architectural solution for structured LDPC code decoding. A widely used approach in LDPC decoders is a layered decoding schedule, which frequently suffers from pipeline data hazards that reduce the throughput. The decoder proposed in the dissertation conveniently incorporates both the layered and the flooding schedules in cases when hazards occur and thus facilitates LDPC decoding without stall cycles caused by pipeline hazards. Therefore, the proposed architecture enables insertion of many pipeline stages, which consequently provides a high operating clock frequency. Additionally, the decoding schedule was optimized for better signal-to-noise ratio (SNR) performance using genetic algorithm. The obtained results show that the proposed decoder achieves great throughput increase and the best HUE when compared with the state of the art for the same SNR performance.

Key words: channel coding, low-density parity-check (LDPC) codes, 5G new radio, flexibility, pipeline, encoder architecture, decoder architecture, layered decoding, hardware usage efficiency, genetic algorithm

Scientific field: Electrical and Computer Engineering

Scientific subfield: Electronics

САДРЖАЈ

Списак слика

СПИСАК ТАБЕЛА

Значење скраћеница и важних симбола

GF	Коначно поље, поље Галоа (енгл. *Galois Field*)
SNR	Однос сигнал-шум (енгл. *Signal-to-Noise Ratio*)
E_s	Енергија по симболу
E_b	Енергија по биту
N_0	Спектрална густина снаге шума
AWGN	Адитивни бели Гаусов шум (енгл. *Additive White Gaussian Noise*)
BSC	Бинарни симетрични канал (енгл. *Binary Symmetric Channel*)
BEC	Бинарни канал са брисањем (енгл. *Binary Erasure Channel*)
BER	Вероватноћа грешке по биту (енгл. *Bit Error Rate*)
FER	Вероватноћа грешке по кодној речи–фрејму (енгл. *Frame Error Rate*)
BLER	Вероватноћа грешке по кодној речи–блоку (енгл. *BLock Error Rate*)
BPSK	Бинарна фазна модулација (енгл. *Binary Phase-Shift Keying*)
QPSK	Квадратурна фазна модулација (енгл. *Quadrature Phase Shift Keying*)
3GPP	Стандардизационо тело (енгл. 3*rd Generation Partnership Project*)
Wi-Fi	IEEE 802.11 фамилија стандарда за бежичне комуникације
WiMAX	IEEE 802.16 фамилија стандарда за бежичне комуникације (енгл. *Worldwide Interoperability for Microwave Access*)
DVB-S/T/C	Стандарди за дигитално емитовање телевизије (енгл. *Digital Video Broadcasting*)
UMTS	Трећа генерација стандарда за мобилне комуникације (енгл. *Universal Mobile Telecommunications System* – 3*G*)
LTE	Четврта генерација стандарда за мобилне комуникације (енгл. *Long-Term Evolution* – 4*G*)
5G NR	Пета генерација стандарда за мобилне комуникације (енгл. 5*G New Radio*)
URLLC	Изузетно поуздане комуникације малог кашњења (енгл. *Ultra-Reliable Low-Latency Communication*)
eMBB	Мобилни сервис у 5G стандарду (енгл. *Enhanced Mobile Broadband*)
LDPC код	Код са проверама парности мале густине (енгл. *Low-Density Parity-Check code*)
QC	Квазицикличан (енгл. *Quasi-Cyclic*)
RA код	Код са акумулацијом бита парности (енгл. *Repeat-Accumulate code*)
IRA код	Нерегуларни код са акумулацијом бит парности (енгл. *Irregular Repeat-Accumulate code*)
BCH код	*Bose–Chaudhuri–Hocquenghem* код (по ауторима)
PEG	Алгоритам конструкције LDPC кодова са прогресивним растом графа кода (енгл. *Progressive Edge-Growth*)
HARQ	Хибридне процедуре са аутоматским ретрансмисијама пакета (енгл. *Hybrid Automatic Repeat-reQuest*)
BG	Основни граф (енгл. *Base Graph*)
$\mathbf{H}$	Контролна матрица линеарног блок кода
$\mathbf{G}$	Генеришућа матрица линеарног блок кода
n	Дужина кодне речи

k	Дужина информационог дела кодне речи
m	Број једначина провера парности
γ	Тежина колона контролне матрице
ρ	Тежина врста контролне матрице
R	Кодни количник
$\mathbf{i}$	Вектор информационих бита
$\mathbf{p}$	Вектор контролних бита (бита парности)
CPM	Кружно померена јединична матрица – циркулант (енгл. *Circulant Permutation Matrix*)
Z	Димензија циркуланта
$\mathbf{P}$	Пермутациона матрица кода/Матрица експонената
k_b	Број група информационих бита
n_b	Број група бита кодне речи, број колона пермутационе матрице, број варијабилних чворова основног графа
m_b	Број врста пермутационе матрице, број контролних чворова основног графа
$\boldsymbol{\lambda}$	Сума кружно померених информационих бита у $GF(2)$
HD	Тврда одлука (енгл. *Hard-Decision*)
SD	Мека одлука (енгл. *Soft-Decision*)
BF	Инвертовање бита (енгл. *Bit Flipping*)
MP	Пропагација порука (енгл. *Message-Passing*)
BP	Пропагација веродостојности (енгл. *Belief-Propagation*)
SPA	Алгоритам сумирања и производа (енгл. *Sum-Product Algorithm*)
MS	Min-Sum алгоритам
OMS	Min-Sum алгоритам са офсетом (енгл. *Offset Min-Sum*)
NMS	Нормализовани Min-Sum алгоритам (енгл. *Normalized Min-Sum*)
LLR	Логаритам односа вероватноћа (енгл. *Logarithm-Likelihood Ratio*)
$M_{v\to c}$	Порука варијабилног чвора (енгл. *Variable-to-check message*)
$M_{c\to v}$	Порука контролног чвора (енгл. *Check-to-variable message*)
ET	Рани завршетак декодовања (енгл. *Early Termination*)
RAW хазард	Хазард података услед превременог читања (енгл. *Read After Write*)
WAR хазард	Хазард података услед превременог уписа (енгл. *Write After Read*)
FPGA	Врста програмабилног чипа (енгл. *Field Programmable Gate Array*)
ASIC	Интегрисано коло посебне намене (енгл. *Application Specific Integrated Circuit*)
LUT	Лукап табела (енгл. *Look-Up Table*)
FF	Флип-флоп (енгл. *Flip-flop*)
BRAM	Блокови RAM у FPGA чипу (енгл. *Block Random Access Memory*)
FIFO бафер	Бафер код кога се прво читају они подаци који су прво уписани (енгл. *First-In-First-Out*)
PS	Степен проточне обраде (енгл. *Pipeline Stage*)
QSN	Мрежа за кружни померај за квазицикличне LDPC декодере (енгл. *Quasi-Cyclic LDPC Shifting Network*)

f_{CLK}	Учестаност сигнала такта (енгл. *clock frequency*)
f_{max}	Максимална учестаност сигнала такта
CPC	Тактова по кодној речи (енгл. *Clocks Per Codeword*)
T	Проток
l	Кашњење (енгл. *latency*)
TSP	Проблем трговачког путника (енгл. *Traveling Salesman Problem*)
GA	Генетички алгоритам (енгл. *Genetic Algorithm*)

1. Увод

Развој савремених дигиталних комуникација заснован је на радовима Клода Шенона (*Claude Shannon*) који је поставио основе теорије информација [1]–[6]. У својим радовима је анализирао пренос информација кроз комуникациони канал, покушавајући да га делимично раздвоји од строге везе са преносом сигнала. Наиме, дотадашњи приступ је подразумевао да изобличење у пренетом сигналу (услед шума, интерференције и сл.) готово сигурно изазива оштећења у примљеним информацијама. Због тога је циљ био што квалитетнији пренос сигнала, тј. повећање односа сигнал-шум (енгл. *Signal-to-Noise Ratio, SNR*), што је резултовало развојем области линеарне филтрације и статистичке детекције [7]. Један од оснивача ове области био је Норберт Винер (*Norbert Wiener*) који је генерализовао хармонијску анализу [8] и, касније, одредио оптимални филтар за случајни сигнал у адитивном шуму [9], [10]. Насупрот овом приступу, Шенон је посматрао саме информације и покушао да одреди под којим је условима могуће пренети информације чак и ако је примљени сигнал изузетно изобличен. Дефинисао је количину информација, информациони проток и капацитет канала, тј. теоријску границу брзине поузданог преноса информација, а затим и повезао капацитет канала са основним комуникационим ресурсима: односом сигнал-шум и расположивим пропусним опсегом [7].

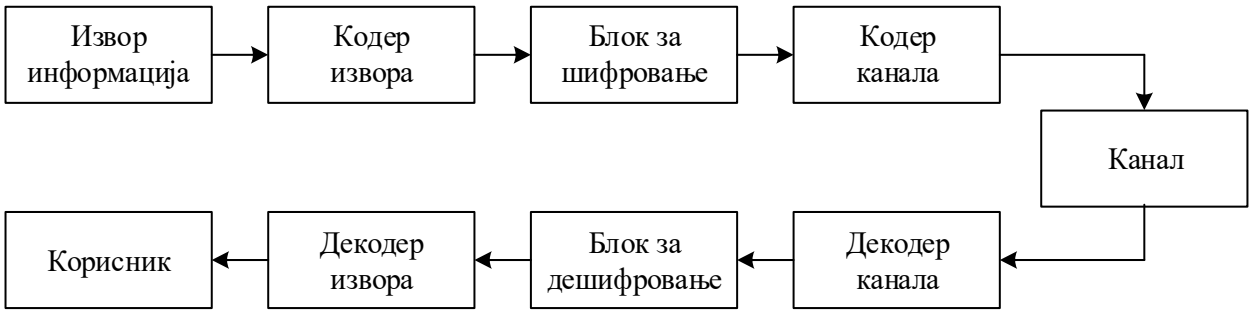
Шенон је у својим радовима поставио две фундаменталне теореме.

Прва Шенонова теорема дефинише теоријски минималан број сигнала (бита) по поруци којим се могу представити информације које емитује извор. У пракси, ефикасну представу порука сигналима обезбеђује статистички кодер (или кодер извора) који се поставља одмах након извора информација, при чему кодови примењени у статистичким кодерима зависе од особина извора. Интересантно, поступци који постижу веома ефикасно статистичко кодовање предложени су убрзо након дефинисања прве Шенонове теореме у радовима самог Шенона [1], Фаноа (*Robert Fano*) [11], Хафмана (*David Huffman*) [12] и касније Фалера (*Newton Faller*) [13], Галагера (*Robert G. Gallager*) [14], Лемпела (*Abraham Lempel*) и Зива (*Yaakov Ziv*) [15] и других [7].

Друга Шенонова теорема бави се поузданошћу преноса информација. По тој теореми, могуће је остварити произвољно малу вероватноћу грешке при преносу све док је информациони проток мањи од капацитета канала. Шенон је доказао да сметње утичу на брзину преноса информација, а не на поузданост. Да би се остварила мала вероватноћа грешке, неопходно је сигналима који носе информације додати редундансу, тј. сигнале који не носе информације, али који могу послужити пријемнику за откривање и исправљање грешака које су настале при преносу. Ову редундансу додаје заштитни кодер (или кодер канала) у коме се примењују кодови који зависе од особина канала. Иако је показао да кôд којим се достиже капацитет канала може да постоји, Шенон у својим радовима није предложио на који начин се такав код може направити [7].

Са становишта теорије информација, комуникациони систем се често представља блок-шемом приказаном на слици 1.1. Поред ефикасног и поузданог преноса, за шта се респективно користе кодер извора и кодер канала на предајној, односно декодер извора и

декодер канала на пријемној страни, често је потребан и безбедан пренос. У те сврхе користе се блокови за шифровање и дешифровање, тј. кодови за очување тајности. Шенон је поставио фундаменталне резултате и на овом пољу [16].



Слика 1.1. Блок-шема комуникационог система са становишта теорије информација

За разлику од техника статистичког кодовања, које су веома брзо након дефинисања прве Шенонове теореме постигле велику ефикасност, било је потребно око пола века да се направе заштитни кодови који омогућавају достизање Шеноновог капацитета. Први кодови који су омогућили значајан прилазак Шеноновој граници били су турбо кодови, предложени крајем XX века [17]. Убрзо након тога, Ричардсон (*Thomas Richardson*) и Урбанке (*Rüdiger Urbanke*) су показали да се кодови са проверама парности мале густине (енгл. *Low-Density Parity-Check, LDPC, codes*) могу приближити Шеноновој граници на мање од десетине децибела [18], [19], док је кôд који јој прилази на 0,0045 dB [20] и практично потврдио другу Шенонову теорему.

Кодове са проверама парности мале густине први је предложио Галагер још 1963. године [21]. Међутим, у време њиховог настанка сматрано је да су сувише комплексни за практичну примену, па су потпуно заборављени све до краја XX века. Тада су их поново открили Меккеј (*David MacKay*) и Нил (*Radford Neal*) и показали да се итеративним декодовањем могу постићи перформансе сличне перформансама тадашњих турбо кодова [22]. Приметили су да је оригинални Галагеров алгоритам декодовања суштински апроксимација алгоритма пропагације веродостојности (енгл. *Belief-Propagation, BP, algorithm*) развијеног у теорији експертских система [23]. Проширењем поменутог итеративног алгоритма декодовања на декодовање помоћу пропагације порука у графовским представама кода [24], остварен је један од кључних резултата за практичну примену LDPC кодова. Све више истраживања је имало фокус на развоју кодова и нових алгоритама кодовања и декодовања, као и на хардверским реализацијама кодера и декодера. То је резултовало применом LDPC кодова за корекцију грешака у меморијама [25] и у великом броју комуникационих стандарда. Неки од њих су IEEE 802.3an (10GBASE-T) [26], IEEE 802.11 (Wi-Fi) [27], IEEE 802.16e (WiMAX) [28], DVB-S2/S2x/T2/C2 (Digital Video Broadcasting) [29], DOCSIS 3.1 (Data Over Cable Service Interface Specification) [30] и 3GPP 5G New Radio (NR) [31]. Ниједна друга класа заштитних кодова није нашла тако широку примену у актуелним стандардима као што је то случај са LDPC кодовима [7].

Савремено друштво захтева све веће брзине преноса информација. Самим тим се све више постављају захтеви за повећањем брзине обраде, протока, система за корекцију грешака који укључују LDPC кодове, као и потреба за њиховом великом флексибилношћу, уз очување добрих перформанси у погледу броја исправљених грешака. Пример су системи из наступајуће пете генерације стандарда за мобилне комуникације (5G NR) који захтевају вишегигабитске протоке [32] и подршку за преко 4000 различитих кодова [31]. У овим изазовима је настала мотивација за рад на дисертацији. Иако је иницијално рад на овој теми имао фокус на реализацији кодера и декодера за кодове из 5G стандарда, највећи број остварених резултата и закључака може се применити и на друге системе који користе LDPC кодове. Дисертација нуди решења за хардверске реализације LDPC кодера и декодера, које

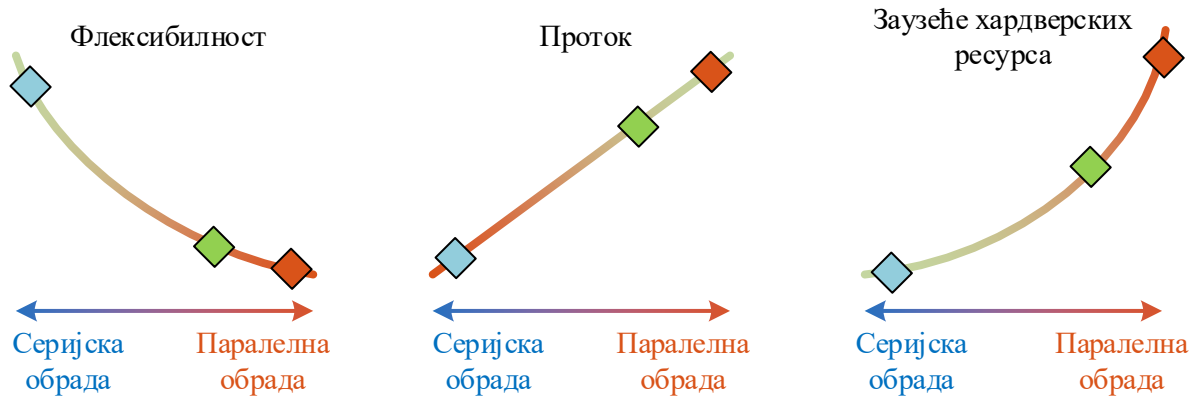
су ефикасне у погледу искоришћења хардверских ресурса, које обезбеђују велике протоке и мала кашњења, уз очување флексибилности какву намећу савремени комуникациони системи.

1.1. Изазови у хардверским реализацијама кодера и декодера LDPC кодова

Од средине деведесетих година прошлог века до данас, системи бежичних комуникација на сваких пет година удесетостручују специфицирани информациони проток, било да се ради о системима комуникација кратког домета (какав је на пример Wi-Fi) или о системима мобилних мрежа (GSM, GPRS, 3G, 4G LTE) [33]. Очекивани протоци у неким применама пете генерације стандарда за мобилне комуникације (5G NR) и савремених система бежичних комуникација кратког домета прелазе вредности од 10 Gb/s [34]. Овакви захтеви стварају велике изазове у пројектовању механизама заштитног кодовања, који су неопходни за поуздане и енергетски ефикасне комуникације, а који су често, у погледу брзине, уско грло целог физичког слоја.

У циљу постизања вишегигабитских протока, поједини делови алгоритама за кодовање и декодовање заштитних кодова морају се одвијати истовремено, у паралели, што захтева велико заузеће рачунских, тј. хардверских, ресурса. Многе класе LDPC кодова пружају могућност за ефикасне хардверске имплементације кодера и декодера и омогућавају велике паралелизме, за разлику од претходно поменутих турбо кодова. Поред протока (енгл. *throughput*) и кашњења (енгл. *latency*), пред пројектанта система заштитног кодовања се постављају додатни захтеви. Најчешћи су перформансе кода и алгоритма декодовања, тј. вероватноћа грешке при различитим односима сигнал-шум или, скраћено, перформансе контроле грешака (енгл. *error correction performance*), затим заузеће хардверских ресурса (енгл. *hardware utilization*) и, све чешће, флексибилност.

Илустрација утицаја нивоа паралелизма у хардверским реализацијама кодера и декодера на флексибилност, проток и заузеће хардверских ресурса приказана је на слици 1.2. Алгоритми за кодовање и декодовање могу се реализовати серијски, што је увек случај ако се користе процесори опште намене. Специјализацијом процесорске јединице за најчешће коришћене операције у алгоритмима кодовања и декодовања може се постићи ефикасна хардверска реализација која пружа највећу флексибилност. Флексибилност се огледа у могућности за измену кода, параметара кодера и декодера, али и самих алгоритама. Заузеће хардверских ресурса је, у том случају, по правилу, веома мало. Међутим, проток архитектура које користе серијску обраду је веома мали, а кашњење изузетно велико, што је готово неприхватљиво у савременим комуникационим системима.



Слика 1.2. Илустрација зависности параметара кодера и декодера LDPC кодова од нивоа паралелизма

Насупрот серијским архитектурама, потпуно паралелне архитектуре остварују велике протоке и мала кашњења. Међутим, такви системи нису флексибилни и најчешће не подржавају више од једног кода. Ипак, ова особина пружа могућност за реализацију специјализоване архитектуре пројектоване за један специфичан кôд која може бити веома ефикасна. Ако се пак жели постићи флексибилност, заузеће хардверских ресурса може драстично да порасте. Честа ситуација је да додавање хардверских ресурса у таквим реализацијама несразмерно мало повећава проток због, између осталог, изражених проблема са рутирањем удаљених блокова, чиме се ефикасност дизајна знатно смањује [35]. Због тога се најчешће у практичним системима користе делимично паралелне архитектуре којима се постижу прихватљиви протоци и флексибилност уз релативно ефикасно искоришћење хардверских ресурса [36].

Уобичајен начин здружене анализе брзине рада (протока и кашњења) и заузећа хардверских ресурса је коришћење њиховог количника. Дељењем оствареног протока заузетом површином на интегрисаном колу или бројем логичких, регистарских и меморијских елемената добија се параметар који се у дисертацији назива ефикасношћу искоришћења хардверских ресурса (енгл. *Hardware Usage Efficiency, HUE*). Оптимизацијом кодера и декодера по овом критеријуму добија се систем највећег протока за најмање заузеће хардверских ресурса. Поред протока, важан временски параметар је и кашњење. Међутим, ако је кашњење једне оптимизоване компоненте прихватљиво за систем, за остваривање произвољно великог протока, много је ефикасније користити више таквих компоненти које раде у паралели, него једну неоптималну која би за исте захтеве заузела много више ресурса. Таква неоптимална компонента би по правилу имала и мању енергетску ефикасност. Ово је један од основних разлога због кога је коришћење делимично паралелних архитектура најчешће у пракси, посебно за информационе блокове велике дужине.

На могућност паралелизације и глобалну архитектуру кодера и декодера највише утиче структура LDPC кодова за које се систем пројектује. Реализација хардвера за случајне кодове је засигурно најкомплекснија и, по сва три поменута параметра, најмање ефикасна. Стога, већина практичних система користи кодове уређене структуре која инхерентно омогућава паралелизацију алгоритама кодовања и декодовања, као и представу параметара LDPC кода погодну за флексибилне реализације. Најпознатија класа кодова која има наведене особине су квазициклични LDPC кодови (енгл. *Quasi-Cyclic, QC, LDPC codes*) [37]–[39]. Чак и кад кодови нису квазициклични, ипак се пројектују тако да се одређеним операцијама могу свести на структуру која подсећа на квазицикличан код.

Квазициклична структура кода омогућава поделу података који представљају кодну реч на групе. Ове групе се могу истовремено читати из меморије, уписивати у меморију и користити за паралелна израчунавања дефинисана алгоритмом кодовања или декодовања. Структура кода захтева да у глобалној архитектури постоје комбинационе мреже за кружни померај одговарајућих група података (енгл. *cyclic shifters*). Број елемената групе одређује колико велику мрежу за кружни померај треба пројектовати. Додатно, у случају флексибилних кодера и декодера, мрежа за кружни померај треба да омогући и померање вектора различите дужине, с обзиром на то да различити кодови захтевају различит број елемената групе. Ефикасна реализација такве мреже није тривијалан задатак и представља један од комплекснијих проблема у пројектовању кодера и декодера за савремене комуникационе системе [40].

Кодери бинарних LDPC кодова генеришу одређене бите парности на основу информационих бита, па су основни процесирајући елементи кодера најчешће логичка кола која изводе операцију „ексклузивно или” („ексили”, енгл. *xor*). С друге стране, процесирајући елементи декодера, блокови у којима се раде израчунавања, у многоме зависе од усвојеног алгоритма декодовања. Структура кода утиче на глобалну архитектуру декодера, али хардверска комплексност процесирајућих елемената и потребна количина меморијских ресурса доминантно зависе од алгоритма декодовања. Очекивано, алгоритми који дају

најбоље перформансе декодовања у погледу броја исправљених грешака су уједно и најкомплекснији за реализацију. Оптималан алгоритам за канал са адитивним белим Гаусовим шумом (енгл. *Additive White Gaussian Noise, AWGN*) у асимптотском случају (за кôд бесконачне дужине) је раније поменути алгоритам пропагације веродостојности (енгл. *belief-propagation*). Процес декодовања се може посматрати као пропагација порука (вероватноћа) између чворова у графовској представи кода. Међутим, овај алгоритам захтева високу тачност израчунавања, што доводи до потребе да се бројеви представљају у формату са покретним зарезом (енгл. *floating-point*) или уз помоћ веома великог броја бита у формату са фиксним зарезом (енгл. *fixed-point*). Због тога се у хардверским реализацијама декодера најчешће користе апроксимације, па су најпознатији алгоритми засновани на тзв. *Min-Sum* апроксимацији [41], [42]. Више детаља о алгоритмима декодовања приказано је у поглављу Кодови са проверама парности мале густине.

Захваљујући итеративном поступку декодовања LDPC кодови имају веома добру способност за корекцију грешака. Међутим, управо итеративна природа алгоритама у великој мери ограничава проток декодера. Стога је дизајн декодера за велике протоке најчешће значајно комплекснији од дизајна кодера. Важна алгоритамска техника која може убрзати декодовање представљена је у [43] и назива се слојевито (енгл. *layered*) декодовање. Наиме, у најчешћој графовској представи LDPC кода, могу се идентификовати две врсте чворова: варијабилни и контролни чворови. Гране графа повезују једну врсту чворова са другом. Оваква представа се, по аутору, назива Танеровим графом [44]. У класичном декодовању помоћу размене порука, сви варијабилни чворови шаљу поруке контролним чворовима симултано. Контролни чворови на основу примљених порука ажурирају своја стања и, као одговор, симултано шаљу своје поруке варијабилним чворовима. Један циклус размене порука од варијабилних ка контролним чворовима и назад представља једну итерацију декодовања. Вредности, које представљају вероватноће да су одговарајући бити кодне речи нуле или јединице, чувају се као стања варијабилних чворова и ажурирају се на крају сваке итерације. Тада се и врши провера да ли вектор бита добијен из ових вероватноћа одговара некој од валидних кодних речи. Ако је добијена валидна кодна реч, процес декодовања се, по правилу, зауставља. У супротном, процес се наставља све док се у некој од наредних итерација не добије валидна кодна реч или док се не стигне до максималног броја предвиђених итерација. У дисертацији ће се за овакав начин декодовања често користити термин „декодовање са симултаном разменом порука" или „симултано декодовање", док се у литератури на енглеском језику често назива „*flooding schedule*" [24]. Насупрот симултаном декодовању, код слојевитог декодовања итерације се могу поделити на подитерације у којима само део варијабилних чворова шаље поруке контролним чворовима са којима је повезан. На тај начин се стања варијабилних чворова ажурирају на крају сваке подитерације, тј. значајно чешће. Ово има за последицу да је декодовање брже, па се због тога и у комерцијално доступним декодерима све више користи слојевито декодовање [45], [46].

Поред алгоритамских техника и паралелизације алгоритама, проток система се може повећати подизањем оперативне учестаности сигнала такта. Максимална вредност учестаности такта одређена је највећим кашњењем логичких елемената које постоји између два регистарска елемента, тј. критичном путањом (енгл. *critical path*). Најчешћа техника којом се повећава максимална учестаност такта је техника проточне обраде (енгл. *pipeline*). Ова техника подразумева поделу критичне путање на више мањих постављањем регистара на погодна места, чиме се смањује кашњење између регистара.

Проточна обрада је неопходна за постизање високих учестаности сигнала такта. Међутим, када год постоји проточна обрада, постоји могућност за појаву конфликата података (енгл. *data hazards*). Најчешћи конфликти података настају када је потребно прочитати податак из меморије пре него што други процес упише ажурну вредност (енгл. *Read After Write, RAW* – читање после уписа, у даљем тексту „конфликти услед превременог читања") [47]. У тој ситуацији је неопходно сачекати упис ажурне вредности како се из

меморије не би прочитали стари подаци који нису валидни. Ово је посебно изражено код декодера који користе слојевито декодовање, јер се вероватноће ажурирају чешће него код симултаног декодовања. Чекање на уписе изазива циклусе паузе и у великој мери може смањити проток. Ако се конфликти игноришу и паузе не праве, број исправљених грешака драстично пада и неретко доводи до ситуације да је декодовање немогуће успешно извршити [48].

Додатно, посебна ситуација може да настане у случају нерегуларних кодова, када се у различитим подитерацијама слојевитог декодовања обрађује различит број варијабилних чворова. Тада постоји могућност да се појави и друга врста конфликата који настају ако неки процесирајући елементи треба да упишу нове податке у меморију или регистре пре него што су други процесирајући елементи прочитали валидне старе вредности (енгл. *Write After Read, WAR* – упис после читања, у даљем тексту „конфликти услед превременог уписа") [47]. Како би декодовање било исправно, и тада је неопходно увести паузе, што додатно смањује проток [49].

Смањење броја циклуса паузе при декодовању представља отворен проблем. У литератури се може наћи више начина за делимично разрешење овог проблема. Први је прерасподела контролне матрице кода у облик који је мање осетљив на међусобне зависности између израчунавања [48], [50]–[52]. У општем случају, овај метод не може уклонити све циклусе паузе и мора се спровести за сваки кôд појединачно. Други приступ подразумева декодовање више кодних речи истовремено [53], [54]. На овај начин декодер врши израчунавања потребна за декодовање неке друге кодне речи док чека на завршетак израчунавања потребних за декодовање прве кодне речи. Уколико се истовремено декодује већи број речи, циклуси паузе се могу у потпуности избећи. Ипак, неопходно је чувати податке о свим кодним речима истовремено, што повећава захтеве за меморијским ресурсима. Додатно, кашњење оваквог система је веће него кашњење система који декодује само једну реч и све хардверске ресурсе усмерава на завршетак декодовања те једне речи. Неке од алгоритамских техника за смањење броја циклуса паузе приказане су у [55] и [56]. Приликом ажурирања стања једне групе варијабилних чворова (уписа нових вредности), техника из [55] захтева и читање тренутног стања на која се додају доприноси контролних чворова. Истовремено, тренутна стања неких других варијабилних чворова се морају читати за израчунавање порука коју ти чворови шаљу својим контролним чворовима. За реализацију су због тога потребне две меморијске банке за стања варијабилних чворова, потпуно истог садржаја, што повећава захтеве за меморијским ресурсима. У приступу приказаном у [56], алгоритам предвиђа наставак декодовања без пауза у израчунавању, али доприносе контролних чворова акумулира у посебној регистарској банци. Ови доприноси се користе за ажурирање стања варијабилних чворова тек када се појави ситуација у којој нема конфликта. На овај начин се избегавају циклуси паузе, али се значајно деградирају перформансе контроле грешака.

1.2. ДОПРИНОСИ ИСТРАЖИВАЊА

Приликом пројектовања декодера постоје два опречна захтева, први: повећати учестаност такта повећањем броја степени проточне обраде и други: смањити број циклуса паузе обрнутим поступком, тј. смањењем броја степени проточне обраде. Идеално решење за овај проблем би било раскидање везе између два захтева или укидање једног од њих. На проток система директно утиче оперативна учестаност сигнала такта. Максимална учестаност зависи од кашњења критичне логичке путање, док се техником проточне обраде логичке путање могу скратити и учестаност тиме повећати. Ако се критична путања подели на велики број делова, тј. ако је број степени проточне обраде велики, максимална учестаност је утолико виша. Међутим, повећање степени проточне обраде изазива већу

зависност међу различитим процесима који раде са истим варијабилним чворовима. Другим речима, број конфликата података, а самим тим и број циклуса паузе, расте.

Дисертација даје алгоритамско решење за слојевито декодовање квазицикличних LDPC кодова које решава конфликте података насталих услед превременог читања без увођења пауза. Алгоритам на погодан начин комбинује слојевито декодовање са концептима симултаног декодовања и раскида зависности између различитих подитерација. Стога је предложени метод у дисертацији назван хибридним декодовањем. Постигнуто је значајно смањење потребних циклуса такта за једну итерацију декодовања. Додатно, укинут је захтев за смањењем броја степени проточне обраде, чиме је омогућено произвољно повећање учестаности сигнала такта.

Алгоритам декодовања је додатно оптимизован за побољшане перформансе контроле грешака. Основа оптимизације је измењени редослед процесирања различитих делова контролне матрице. Налажење оптималног редоследа процесирања се може разумети као познати проблем трговачког путника који је погодно решавати генетичким алгоритмом [57], који је искоришћен за оптимизацију и у дисертацији. Хибридно декодовање и његова оптимизација пружају могућност за велико повећање протока декодера у поређењу са слојевитим декодовањем. Као пример, кодови из 5G NR стандарда се, у зависности од кодног количника, могу декодовати између 30% и 109% брже него слојевитим декодовањем уз задржавање истих перформанси контроле грешака.

У дисертацији је предложена флексибилна хардверска архитектура декодера која подржава велики број кодова и дужина кодних речи, заснована на наведеном алгоритамском решењу. Поред потпуног уклањања циклуса паузе насталих због конфликата података услед превременог читања, предложено је хардверско решење које омогућава декодовање без конфликата података услед превременог уписа који настају при декодовању неких нерегуларних кодова. Реализација наведене архитектуре на FPGA програмабилном чипу (енгл. *Field Programmable Gate Array*), у поређењу са претходно публикованим резултатима, даје најбољу ефикасност искоришћења хардверских ресурса.

Поред декодера, дисертација се бави и хардверском реализацијом кодера за квазицикличне кодове из 5G NR стандарда. Постигнута је велика ефикасност искоришћења хардверских ресурса за флексибилни кодер који подржава све кодове предвиђене стандардом. Кључни хардверски блок кодера је флексибилна мрежа за кружни померај једног великог блока података, када је то потребно, али и паралелни померај више мањих блокова. Тако се за кодове чије су кодне речи краће, може постићи већи паралелизам него што је инхерентно предвиђено самим кодом. Генетички алгоритам је искоришћен и на овом месту да нађе оптималан редослед процесирања са циљем минимизације времена кодовања. Поред примене за кодове из 5G NR стандарда, показано је да се остварени доприноси могу применити и за кодовање других LDPC кодова.

Кључни доприноси дисертације представљени су у следећим публикацијама:

[49] **V. L. Petrović**, M. M. Marković, D. M. El Mezeni, L. V. Saranovac, and A. Radošević, "Flexible High Throughput QC-LDPC Decoder with Perfect Pipeline Conflicts Resolution and Efficient Hardware Utilization," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 12, pp. 5454–5467, Dec. 2020, doi: 10.1109/TCSI.2020.3018048 (**M21, IF 2019: 3.318**)

[58] **V. L. Petrović**, D. M. El Mezeni, and A. Radošević, "Flexible 5G New Radio LDPC Encoder Optimized for High Hardware Usage," *Electronics*, vol. 10, no. 9, p. 1106, May 2021, doi: 10.3390/electronics10091106 (**M22, IF 2019: 2.412**)

[59] **V. L. Petrović** and D. M. El Mezeni, "Reduced-Complexity Offset Min-Sum Based Layered Decoding for 5G LDPC Codes," in *Proceedings on 2020 28th Telecommunications Forum (TELFOR)*, Belgrade, Serbia, Nov. 2020, pp. 109–112, doi: 10.1109/TELFOR51502.2020.9306590 (**M33**)

[60] **V. L. Petrović** and D. M. El Mezeni, "Reduced-Complexity Offset Min-Sum Check Node Unit for Layered 5G LDPC Decoder," *Telfor Journal*, vol. 13, no. 1, pp. 7–12, Jul. 2021, doi: 10.5937/telfor2101007P (**M53**)

1.3. СТРУКТУРА ДИСЕРТАЦИЈЕ

Материја представљена у дисертацији приказана је у пет поглавља.

Увод у проблеме који су решавани у дисертацији приказан је у првом поглављу.

Друго поглавље описује основне појмове везане за кодове са проверама парности мале густине, а који се користе у дисертацији. Описане су најважније класе LDPC кодова, њихове особине и неки методи конструкција. Посебна пажња је посвећена стандардизованим кодовима који се примењују у пракси. Након тога је дат преглед најчешће коришћених метода кодовања LDPC кодова са освртом на њихову рачунску комплексност. На крају, описано је декодовање LDPC кодова. Дат је преглед неколико метода који раде са једнобитним информацијама (тврде информације) и метода који користе вишебитне информације (меке информације). Алгоритми декодовања упоређени су у погледу перформанси контроле грешака и рачунске комплексности. На крају поглавља описан је концепт слојевитог декодовања коришћен за реализацију декодера у овој дисертацији.

У трећем поглављу приказана је ефикасна хардверска реализација кодера за квазицикличне кодове првенствено намењена за 5G NR стандард. Описан је алгоритам за ефикасно кодовање LDPC кодова из 5G NR стандарда. Затим су анализиране архитектуре за хардверску реализацију наведеног алгоритма различитих нивоа паралелизма. Извршено је поређење ових архитектура у погледу остваривог протока и заузећа хардверских ресурса. На основу датог поређења, предложен је нови метод кодовања са променљивим бројем паралелних операција у зависности од дужине кода. Динамичка промена броја паралелних операција подразумева да се за дугачке кодове истовремено процесира само један блок информационих бита по циклусу такта, јер је, због његове велике дужине, за процесирање потребна релативно велика количина хардверских ресурса, док се за краће кодове, код којих је дужина блокова информационих бита мала, паралелно процесира више њих. На тај начин хардверски ресурси се ефикасније користе, па је могуће повећати проток за кодове мале и средње дужине. Затим је приказан метод за оптимизацију редоследа процесирања заснован на генетичком алгоритму којим се постиже минимално трајање кодовања. Након тога, дата је хардверска архитектура којом се реализује описано кодовање. Предложена је нова архитектура флексибилне мреже за кружни померај која је кључни блок у целој архитектури кодера. Приказани су резултати протока, кашњења и ефикасности искоришћења хардверских ресурса за нову архитектуру кодера, али и за друге архитектуре присутне у литератури. Показано је да је кашњење хардверске реализације предложене у дисертацији веома мало, а ефикасност искоришћења хардверских ресурса значајно већа од раније коришћених изузетно паралелних реализација. Тиме је потврђено да је за постизање великог протока много смисленије користити неколико, по ефикасности оптималних кодера, који раде паралелно, него један изузетно паралелизован кодер, код кога је ефикасност искоришћења хардверских ресурса мала. Највећи део приказаних резултата у овом поглављу објављен је у раду [58].

У четвртом поглављу описан је нови декодер квазицикличних LDPC кодова који суштински врши слојевито декодовање, али у одређеним ситуацијама прелази на симултано декодовање које се обавља на малом субграфу графа целог кода. На почетку поглавља дат је детаљан опис конвенционалних архитектура за слојевито декодовање. Описани су параметри који утичу на проток и приказани репрезентативни временски дијаграми који илуструју рад декодера и потребу за увођењем пауза са циљем избегавања хазарда података. Након тога описана су нека алгоритамска решења присутна у литератури којима се могу избећи циклуси паузе, а затим и ново решење предложено у овој дисертацији које је названо хибридним декодовањем. Описана је хардверска архитектура декодера који користи хибридно декодовање, као и модула за израчунавање синдрома и терминацију декодовања када се добије валидна кодна реч. Додатно, дато је решење за хазарде података услед превременог уписа који настају у процесорима контролних чворова и захтевају увођење додатних циклуса паузе. Након описа хардверске архитектуре приказан је алгоритам за оптимизацију

редоследа процесирања којим се добијају побољшане перформансе контроле грешака. Оптимизациони поступак је заснован на генетичком алгоритму. Приказане су перформансе контроле грешака за оптимизовано хибридно декодовање и декодовање са оригиналним редоследом процесирања, а извршено је и поређење са перформансама слојевитог и симултаног декодовања. Резултати показују одређени губитак у перформансама контроле грешака хибридног декодера у поређењу са перформансама декодера који користи слојевито декодовање на истом броју итерација. Међутим, губитак се може надокнадити увођењем додатних итерација декодовања, па је затим урађена здружена анализа перформанси контроле грешака и протока. Поред анализе перформанси контроле грешака дато је поређење трајања једне итерације хибридног декодовања и слојевитог декодовања. У том погледу, показана је супериорност хибридног декодовања које доприноси томе да се и поред увођења додатних итерација и даље добија значајно већи проток за исте или боље перформансе контроле грешака у поређењу са слојевитим декодовањем. Коначно, приказани су резултати имплементације на FPGA платформи и поређење са референтним радовима из литературе. Највећи део поменутих метода и резултата објављен је у раду [49].

Поред наведеног, урађена је и опсежнија анализа перформанси контроле грешака пројектованог декодера за кодове из 5G NR стандарда. Перформансе су испитане за 12 различитих кодова. Показано је да за неке кодове декодер не може да постигне вредности за вероватноћу грешке по кодној речи мање од 10^{-5}, што је захтев у одређеним будућим применама дефинисаним 5G стандардом [61]. Проблем је решен повећањем динамичког опсега одређених вредности у хардверској реализацији, тј. повећањем броја бита за представу бројева. Како би се избегло значајно повећање заузећа хардверских ресурса, предложено је да се за одређене вредности користи нелинеарна квантизација, чиме је добијен декодер значајно бољих перформанси контроле грешака чије је заузеће хардверских ресурса упоредиво са претходном реализацијом. На крају излагања о хардверској реализацији декодера приказан је један покушај смањења комплексности декодовања 5G NR LDPC кодова и архитектура процесорске јединице којом се постиже смањена комплексност. Основни резултати из ове теме приказани су у горенаведеним радовима [59] и [60], а у овој дисертацији решење је проширено на процесирајуће елементе којима се разрешавају хазарди података услед превременог уписа.

Закључак дисертације и даљи правци истраживања дати су у петом поглављу.

2. КОДОВИ СА ПРОВЕРАМА ПАРНОСТИ МАЛЕ ГУСТИНЕ

2.1. ОСНОВНИ ПОЈМОВИ

Заштитни кодер пре слања информација кроз комуникациони канал додаје редундансу сигналима који носе информације. Та редунданса је представљена сигналима који не носе информације, али које декодер користи за потребе детекције и исправљања грешака насталих при преносу. У многим практичним применама, информације на улазу у кодер су представљене бинарном секвенцом (секвенцом нула и јединица), где сваки елемент секвенце представља један информациони бит. На основу информационе секвенце, кодер генерише редундантне заштитне бите (контролне бите). Ако је дужина информационе секвенце коначне дужине k и ако се, на основу те секвенце, генерише коначан број контролних бита m, чиме се формира кодна реч дужине $n = k + m$, скуп свих 2^k кодних речи назива се блок кодом. При том се дефинише кодни количник кода као $R = k / n$. Блок кôд код кога је кодна реч дужине n, а информациона секвенца дужине k, најчешће се означава уређеним паром (n,k). Уобичајено је да се контролни бити код бинарних кодова генеришу проверама парности, односно непарности, броја јединица у m различитих подсеквенци информационе секвенце. Због тога се контролни бити још називају и битима парности (енгл. *parity bits*) који имају вредност „1” ако је број јединица у посматраној подсеквенци информационих бита непаран, а вредност „0” ако је број јединица у посматраној информационој подсеквенци паран.

С обзиром на то да су код бинарних кодова све кодне речи сачињене од свега два елемента, може се рећи да су све кодне речи елементи векторског простора над коначним пољем *GF*(2). Овај векторски простор садржи укупно 2^n могућих секвенци дужине n, а блок код је скуп свега 2^k секвенци дефинисаних пресликавањем информационих секвенци у одговарајуће кодне речи. Ако кодови нису бинарни, онда се информације представљају симболима из већег скупа, тј. тада су кодне речи састављене од кодних симбола који припадају коначном пољу $GF(q)$ за $q > 2$. У општем случају, за произвољно q, ако скуп кодних речи представља потпростор векторског простора над $GF(q)$, онда се такав код назива линеарним блок кодом. LDPC кодови спадају у класу линеарних блок кодова [7].

Најчешће коришћени LDPC кодови су бинарни, па је и фокус дисертације усмерен на дизајн кодера и декодера за бинарне кодове. Због тога ће се и у даљем тексту сви информациони и контролни симболи називати битима. Међутим, постоје значајни теоријски [62] и практични резултати [63] на пољу небинарних LDPC кодова (енгл. *non-binary LDPC codes*), који имају потенцијал за исправљање пакетских грешака и повећање протока система заштитног кодовања, па је очекивана њихова примена у будућности [64].

Линеарни блок кодови се описују генеришућом матрицом $\mathbf{G}$ димензија $k \times n$, која у потпуности дефинише сва пресликавања из информационих секвенци у кодне речи. Врсте

генеришуће матрице су линеарно независни кодни вектори (кодне речи) који представљају базу кодног потпростора и из којих се генеришу сви могући кодни вектори. Кодне речи $\mathbf{x} = [x_1\ x_2\ \cdots\ x_n]$ могу се добити множењем секвенце информационих бита $\mathbf{i} = [i_1\ i_2\ \cdots\ i_k]$ генеришућом матрицом:

$$\mathbf{x} = \mathbf{i} \cdot \mathbf{G}. \tag{1}$$

Сва израчунавања (сабирања и множења) су у коначном пољу $GF(2)$.

Највећи број практичних LDPC кодова су систематски кодови у чијим су кодним речима информациони бити груписани и неизмењени, а контролни бити само додати на секвенцу информационих бита. Неретко су информациони бити на почетку кодне речи и тада се кодна реч може представити као

$$\mathbf{x} = [\mathbf{i} \quad \mathbf{p}], \tag{2}$$

где секвенца $\mathbf{p} = [p_1\ p_2\ \cdots\ p_m]$ представља низ контролних бита (бита парности).

Након преноса кодне речи кроз канал, на одређеним позицијама се јављају грешке и на страни пријемника добија се измењена секвенца. Симболично се ово може приказати са

$$\mathbf{y} = \mathbf{x} + \mathbf{e}, \tag{3}$$

где је $\mathbf{y} = [y_1\ y_2\ \cdots\ y_n]$ вектор примљених бита, а $\mathbf{e} = [e_1\ e_2\ \cdots\ e_n]$ вектор којим се моделује унос грешака у каналу. У једноставном декодеру је потребно проверити да ли примљена секвенца одговара некој кодној речи. Ова провера је заснована на следећим чињеницама. Векторском потпростору кода се може придружити њему ортогоналан векторски простор чији су сви вектори ортогонални свим векторима кодног потпростора. Тада су и вектори из базе ортогоналног потпростора (базисни вектори) ортогонални свим векторима из кодног потпростора. То значи да је скаларни производ сваког базисног вектора ортогоналног потпростора са било којом кодном речју кодног потпростора једнак нули. Стога би се провера да ли примљена кодна реч припада коду могла урадити израчунавањем скаларног производа вектора примљене кодне речи са свим базисним векторима поменутог ортогоналног потпростора. Базисни вектори ортогоналног потпростора сложени у матрицу $\mathbf{H}$, димензија $m \times n$, и генеришућа матрица морају задовољити следећу једнакост:

$$\mathbf{G} \cdot \mathbf{H}^{\mathrm{T}} = \mathbf{0}. \tag{4}$$

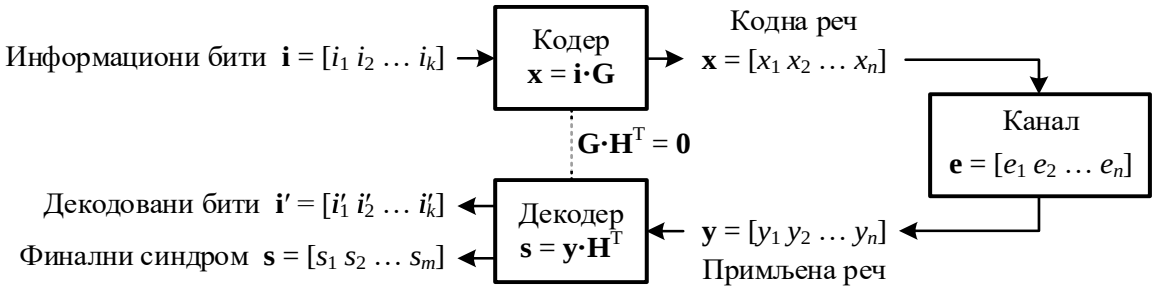
Матрица $\mathbf{H}$ се назива контролном матрицом или матрицом провера парности (енгл. *parity-check matrix*) и одређује се решавањем једначине (4). Коначно, провера се врши израчунавањем синдрома (енгл. *syndrome*):

$$\mathbf{s} = \mathbf{y} \cdot \mathbf{H}^{\mathrm{T}}, \tag{5}$$

који мора бити једнак нула-вектору ако је примљена кодна реч. Ако није било грешака при преносу примљена кодна реч ће одговарати послатој и оваква детекција исправног преноса ће бити тачна. Међутим, постоји могућност да грешке преведу послату кодну реч у неку другу кодну реч. Вероватноћа да се ово догоди зависи од тога на који начин је пројектован код и одређена је минималним Хеминговим растојањем, тј. бројем јединица у секвенци грешака која једну кодну реч преводи у другу. Очекивано, добри кодови по правилу имају велико минимално Хемингово растојање [7].

На слици 2.1 је илустрован уведени формализам кроз блок-шему. Поред израчунавања синдрома, декодер треба да исправи грешке ако добијени синдром није једнак нула-вектору. За исправљање грешака се такође користи контролна матрица. Сваки алгоритам декодовања креће од примљене секвенце коју ажурира по одређеним правилима, најчешће на основу

провера колико је тренутна секвенца бита (процена речи $\mathbf{v} = [v_1\ v_2\ \cdots\ v_n]$) далеко од валидне кодне речи. Поступак може бити итеративан што најчешће даје и најбоље резултате. Након завршеног декодовања добијају се декодовани бити од којих су кориснику најчешће једино значајни информациони бити. С обзиром на то да се не може са сигурношћу тврдити да је секвенца декодованих информационих бита иста као и послата, за њу се уводи посебна ознака $\mathbf{i}'$. Кориснику је често од интереса да зна да ли је декодована секвенца бита заиста валидна кодна реч, па због тога декодер генерише и вредност синдрома израчунату на крају декодовања.



Слика 2.1. Поједностављена блок-шема заштитног кодовања и преноса кроз канал

Важно је напоменути да примљена секвенца на излазу канала не мора нужно бити бинарна. На основу вредности примљених сигнала и познавања услова канала (нпр. односа сигнал-шум), демодулатор може дати секвенцу одређених вероватноћа да су бити примљене речи нуле или јединице. Декодер уместо тврдих одлука (енгл. *hard-decision*) о вредностима бита може користити меке одлуке (енгл. *soft-decision*), које представљају поменуте вероватноће, и на тај начин постићи боље перформансе контроле грешака [65]. Декодовање засновано на тврдим одлукама је рачунски најмање комплексно, али даје и значајно лошије перформансе од декодовања заснованог на меким одлукама. Више детаља о овој теми дато је у потпоглављу 2.4.

Скаларни производ примљене секвенце бита и контролне матрице заправо представља провере парности броја јединица у подсеквенци примљене речи. На пример, ако је дата контролна матрица

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}, \tag{6}$$

вредности синдрома се израчунавају следећим једначинама (контролним сумама):

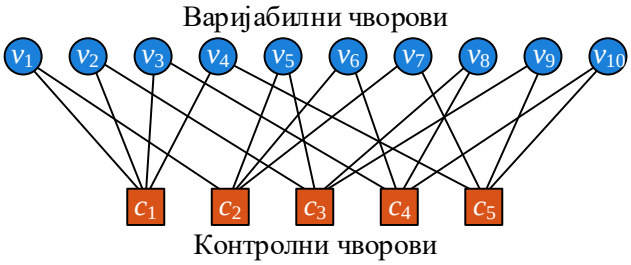
$$\begin{aligned} s_1 &= y_1 + y_2 + y_3 + y_4 \\ s_2 &= y_1 + y_5 + y_6 + y_7 \\ s_3 &= y_2 + y_5 + y_8 + y_9 \\ s_4 &= y_3 + y_6 + y_8 + y_{10} \\ s_1 &= y_4 + y_7 + y_9 + y_{10} \end{aligned} . \tag{7}$$

Свака колона контролне матрице одговара једном биту речи, а свака врста једној контролној суми. Из једначина (7) лако је закључити да је синдром једнак јединици ако је број јединица

међу члановима који улазе у контролну суму непаран, а нули ако је паран, па одатле израз „провера парности”.

Декодовање је најчешће значајно комплекснији процес од кодовања, поготово ако се кодови декодују итеративно. Због тога је погодно да контролна матрица буде ретка (енгл. *sparse*) како би се смањила комплексност израчунавања синдрома и процеса декодовања. Ово је уочио Галагер и, вођен тим принципом, предложио LDPC кодове који су по малој густини контролне матрице и добили име [21].

Танер (*Robert Michael Tanner*) је 1981. предложио графовски опис структуре LDPC кодова [44]. Најчешће коришћена представа је помоћу бипартитног графа (приказаног на слици 2.2) у коме се могу идентификовати две врсте чворова. Варијабилни (енгл. *variable*) или симболски чворови представљају бите тренутне процене речи $\mathbf{v}=\left[v_1\ v_2\ \cdots\ v_n\right]$, док контролни (енгл. *check*) чворови представљају контролне суме чије су тренутне вредности обележене секвенцом $\mathbf{c}=\left[c_1\ c_2\ \cdots\ c_m\right]$. Веза између варијабилног и контролног чвора постоји ако је у одговарајућој колони и врсти контролне матрице јединица. Танеров граф са слике 2.2 одговара контролној матрици из (6). Потребно је још напоменути да се варијабилни чворови који су повезани на исти контролни чвор називају суседним чворовима и обрнуто.



Слика 2.2. Представа LDPC кода помоћу Танеровог графа

У контролној матрици кода из (6) број јединица у свакој врсти је једнак, што важи и за број јединица у свакој колони. Слично, и у представи истог кода Танеровим графом са слике 2.2, из сваког контролног чвора креће једнак број грана и, такође, из сваког варијабилног чвора креће исти број грана. Код са оваквом особином се назива регуларним LDPC кодом (енгл. *regular LDPC code*). Бројеви јединица у врстама се у литератури називају тежинама врста ρ (енгл. *row weights*), а бројеви јединица у колонама тежинама колона γ (енгл. *column weights*). Ако се говори о регуларном коду, често се користи и термин (γ,ρ) -регуларан код. Кодови код којих све тежине врста или све тежине колона нису једнаке називају се нерегуларним LDPC кодовима (енгл. *irregular LDPC codes*) [7]. Расподела тежина врста и тежина колона назива се профил нерегуларности кода или профил кода (енгл. *irregularity profile*) [66]. Због веће слободе при дизајну кода, нерегуларни кодови по правилу могу остварити боље перформансе контроле грешака од регуларних кодова [19].

За регуларне кодове, тежине врста и колона се лако могу довести у везу са кодним количником. С обзиром на то да у свакој од m врста има ρ јединица и у свакој од n колона има γ јединица, важи да је $\rho m=\rho\left(n-k\right)=\gamma n$, па се одатле добија

$$R=\frac{k}{n}=\frac{n-\dfrac{\gamma n}{\rho}}{n}=1-\frac{\gamma}{\rho}. \tag{8}$$

Добијени израз се може користити и за нерегуларне кодове, али се за тежине врста и колона морају узети просечне вредности, тј.

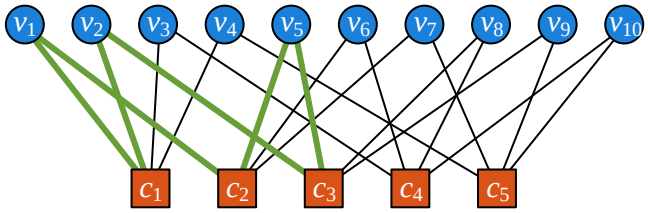
$$R = \frac{k}{n} = 1 - \frac{\bar{\gamma}}{\bar{\rho}}. \tag{9}$$

Потребно је још поменути да перформансе контроле грешака зависе од дужине кодне речи. Дуги кодови по правилу дају боље перформансе од кратких. Међутим, системи који користе дуге кодне речи имају и велико кашњење, па се дуги кодови користе тамо где кашњење није критичан захтев, као што је то у сателитским комуникацијама. Тамо где је кашњење важан критеријум, користе се краћи кодови, али се недостаци у погледу перформанси контроле грешака могу донекле надокнадити коришћењем хибридних процедура са аутоматским ретрансмисијама пакета [67] (енгл. *Hybrid Automatic Repeat-reQuest, HARQ*) [7].

2.2. Конструкција и структура LDPC кодова

LDPC кодови се конструишу тако да им контролна матрица буде ретка са циљем да се комплексност декодовања што је могуће више смањи. Због тога се процес конструкције кода своди на конструкцију контролне, а не генеришуће матрице што је случај са многим другим класама линеарних блок кодова. Контролна матрица се увек пројектује поштујући неке предефинисане критеријуме као што су дужина кода, профил кода и кодни количник. Поред ових критеријума, важан захтев при конструкцији LDPC кодова је да у Танеровом графу не постоје циклуси (петље) мале дужине, а посебно циклуси дужине четири (енгл. 4-*girth-cycle*) који доводе до веома лоших перформанси [7], [65]. У примеру (6), циклус минималне дужине (енгл. *girth*) је шест, што је илустровано на слици 2.3.

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$



Слика 2.3. Циклус минималне дужине у графу кода

Позиције јединица у контролној матрици одређују перформансе контроле грешака, али и могућност за ефикасне хардверске реализације кодера и декодера. У наредним одељцима укратко су описани неки методи конструкције контролне матрице. На крају потпоглавља дат је преглед кодова из неких важних комуникационих стандарда који су коришћени за евалуацију кодера и декодера у оквиру ове дисертације.

2.2.1. Случајни и псеудослучајни LDPC кодови

Случајни кодови се пројектују насумичним постављањем јединица у матрицу предефинисане величине. Димензије контролне матрице зависе од дужине кода и од кодног количника. Насумично постављање јединица врши се поштујући раније поменута ограничења (циклусе минималне дужине, тежине врста и колона, итд.). Јединице се постављају једна по једна на празна места у матрици све док се матрица не попуни жељеним бројем јединица. Уколико није могуће испоштовати задата ограничења приликом постављања неке од јединица, она се одбацује и нова јединица се поставља на друго место [68], [69].

Једно од задатих ограничења може бити и линеарна независност врста у контролној матрици, не би ли оне формирале базу векторског потпростора. Међутим, уз сва претходна ограничења, питање је колико би времена било потребно оваквом методу да направи матрицу која задовољава све критеријуме и да ли ју је уопште и могуће направити. Ипак, допуштањем да врсте контролне матрице буду линеарно зависне, генерисао би се код за чији кодни количник се једино може тврдити да је већи или једнак $1-\gamma/\rho$ [65]. У том случају би се крајњим избацивањем линеарно зависних врста добила коректна контролна матрица линеарног блок кода $\mathbf{H}_{\mathrm{LN}}$ чији би кодни количник заиста био $R=1-\bar{\gamma}_{\mathrm{LN}}/\rho$, где индекс „LN" означава да је реч о просечној тежини колона матрице $\mathbf{H}_{\mathrm{LN}}$. Треба рећи да, иако матрица $\mathbf{H}_{\mathrm{LN}}$ не мора имати једнак број јединица у свим колонама, овај кôд се и даље сматра регуларним јер је оригинална матрица била матрица регуларног кода. Резултат декодовања помоћу синдрома је исти било да се користи оригинална матрица са линеарно зависним врстама или матрица $\mathbf{H}_{\mathrm{LN}}$ [7].

Како би се лакше задовољили раније поменути критеријуми, Галагер је оригинално предложио псеудослучајну конструкцију контролне матрице LDPC кода [21]. Он је искористио чињеницу да контролна матрица не мора нужно имати линеарно независне врсте и предложио метод који најчешће даје регуларне кодове веома добрих перформанси контроле грешака. Матрица се састоји од γ подматрица димензија $(m/\gamma)\times n$ поређаних једна испод друге. Прва подматрица се генерише псеудослучајно, водећи рачуна о томе да ниједна колона нема више од једне јединице и да се у свакој врсти налази тачно ρ јединица. Све преостале подматрице формирају се случајним пермутацијама колона прве подматрице. При томе се води рачуна да свака врста подматрице сме да има највише једну јединицу на истој позицији као иста врста било које друге подматрице. Последње ограничење је важно јер се њиме засигурно избегавају циклуси дужине четири. Добијена контролна матрица нема увек линеарно независне колоне, па је стварни кодни количник кода одређен са $R=1-r(\mathbf{H})/n$, где је са $r(\mathbf{H})$ означен ранг контролне матрице.

Сличан метод конструкције LDPC кода предложили су Меккеј и Нил у радовима [70] и [68]. Контролна матрица се конструише спајањем колона са случајно распоређених γ јединица, при чему се води рачуна о томе да је број јединица у врстама што је могуће више униформан и да тежи ρ. Дозвољава се мала нерегуларност кода уз додатна ограничења да део контролне матрице мора бити несингуларна подматрица, чиме се постиже једноставније кодовање. Циклуси дужине четири се елиминишу избацивањем колона због којих настају, чиме се добија мало мањи кодни количник од жељеног.

Контролна матрица случајних и псеудослучајних кодова није структурирана, па је реализација и кодовања и декодовања за кодове средње и велике дужине изузетно сложена. Додатно, поменути методи нису детерминистички, па се стварни кодни количник не може унапред одредити, а минимално Хемингово растојање је или релативно мало или га је веома тешко одредити.

2.2.2. LDPC КОДОВИ ЗАСНОВАНИ НА КОНАЧНИМ ГЕОМЕТРИЈАМА

Како би омогућили детерминистички приступ у пројектовању LDPC кода, Ку, Лин и Фасорије су предложили алгебарску конструкцију кода засновану на коначним геометријама [71]. Укратко, коначна геометрија је геометријски систем који има коначан број тачака и може се конструисати алгебарски, полазећи од векторских простора над коначним Галоа пољима [72]. Без улажења у детаље метода, овде је поменута само основна идеја и карактеристичне особине добијених кодова.

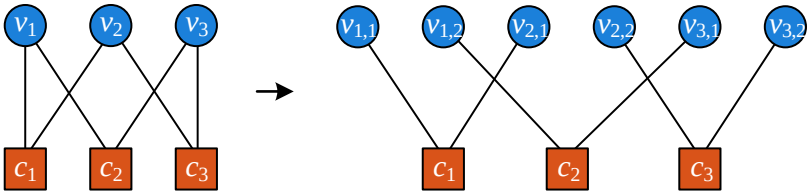
Контролна матрица кода доводи се у везу са коначном геометријом која представља скуп тачака, скуп посебно дефинисаних линија и која садржи информације о томе која тачка се налази на којој линији. Показано је да постоји потпуно пресликавање између позиција јединица у контролној матрици и позиција тачака и линија у коначној геометрији, чиме се поступак пројектовања контролне матрице своди на конструкцију коначне геометрије.

Коначна геометрија $\boldsymbol{G}$ која одговара контролној матрици LDPC кода се састоји од n тачака и J линија које морају задовољити следеће структуралне особине:

- свака линија садржи тачно ρ тачака;
- било које две тачке су повезане једном и само једном линијом;
- у свакој тачки се сече тачно γ линија;
- две линије су или паралелне (тј. не садрже заједничку тачку) или се секу у једној и само једној тачки.

Из наведених особина се може закључити да свака линија одговара једној контролној суми у коју улази ρ вредности варијабилних чворова. Такође, свака тачка одговара једном варијабилном чвору у Танеровом графу. Контролна матрица $\mathbf{H}=\left[h_{i,j}\right]$ је димензија $J\times n$, а елемент матрице $h_{i,j}$ је једнак јединици ако i-та линија коначне геометрије $\boldsymbol{G}$ садржи j-ту тачку те коначне геометрије. У супротном, елемент $h_{i,j}$ једнак је нули. Добијена матрица не мора имати линеарно независне врсте, па је, као и код случајних кодова код којих врсте нису линеарно независне, стварни кодни количник одређен њеним рангом $R=1-r\left(\mathbf{H}\right)/n$.

Кодови добијени на овај начин су јасно структурирани што у великој мери олакшава кодовање и декодовање. Минимално Хемингово растојање је ограничено са доње стране и засигурно је веће од γ. Због дефинисаних особина коначне геометрије, не постоје циклуси дужине четири, али постоје циклуси дужине шест. Међутим, могуће је разбити све краће циклусе техником дуплирања свих варијабилних чворова који формирају циклус, што је илустровано на слици 2.4. Тиме се увећава број колона контролне матрице и добија кôд већег кодног количника, али значајно бољих перформанси. Слично, циклуси се могу разбити и дуплирањем контролних чворова, чиме се задржава дужина кодне речи, али смањује кодни количник [71].



Слика 2.4. Разбијање циклуса дужине шест повећањем броја варијабилних чворова [71]

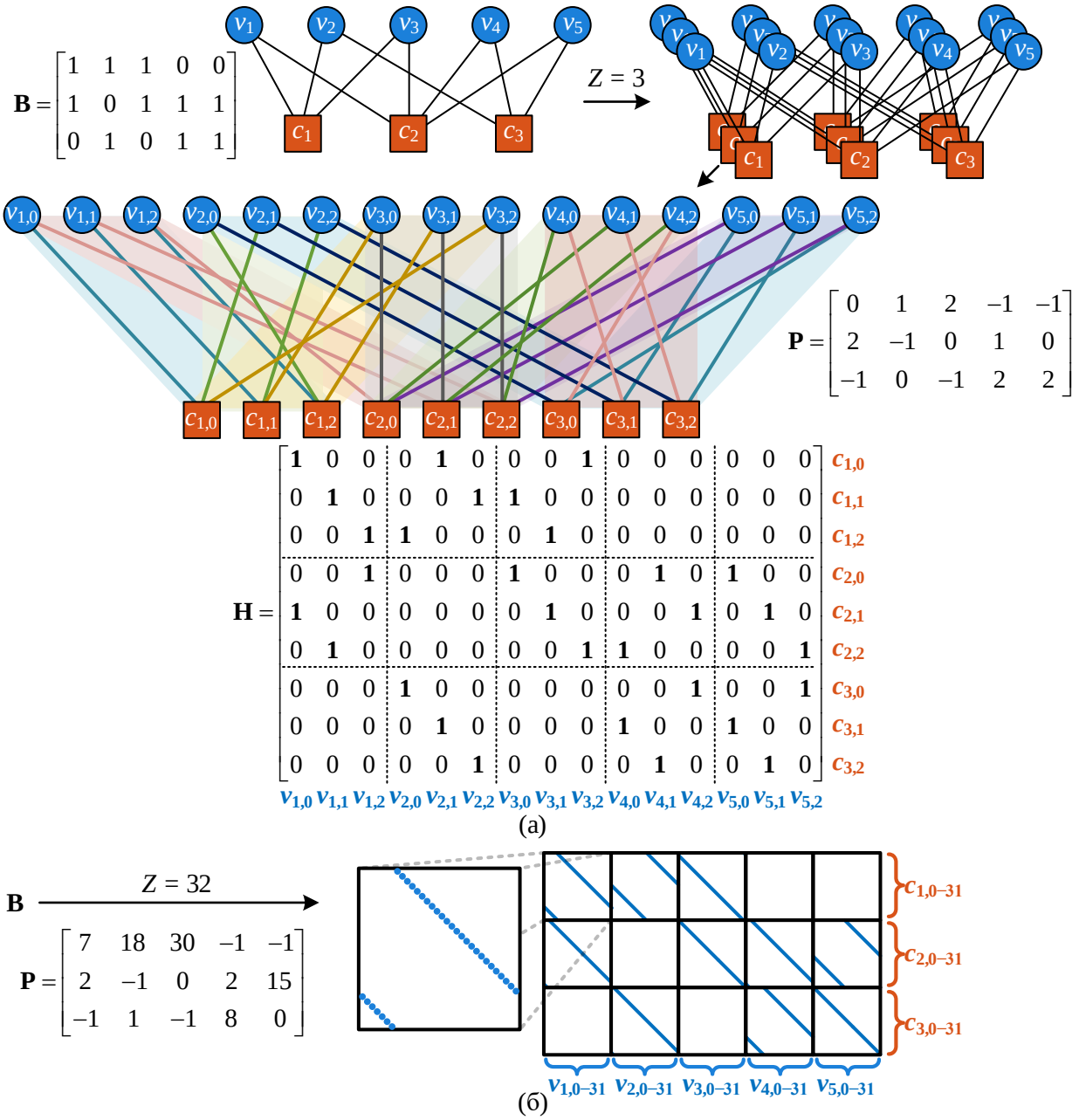
Овај метод конструкције најчешће не може дати велики број кодова и није погодан за конструкцију кодова чије су кодне речи дугачке. Разлог за то је експоненцијални раст броја јединица у контролној матрици са повећањем дужине кода чиме се повећава комплексност и кодовања и декодовања [65].

2.2.3. Квазициклични LDPC кодови

За практичне примене, вероватно најзначајнија класа LDPC кодова која је примењена у великом броју комуникационих стандарда јесу квазициклични LDPC кодови (енгл. *Quasi-Cyclic, QC, LDPC codes*) [37], [38], [39], [73]. Контролна матрица квазицикличног кода састоји се од квадратних подматрица, димензија $Z\times Z$, које могу бити нула-матрице или одређени број пута кружно померене јединичне матрице. Ови квадратни блокови ће се у

даљем тексту називати циркулантима (од енглеског *Circulant Permutation Matrix, CPM* [39]). Између осталих, и кодови засновани на коначним геометријама су циклични или квазициклични.

Пројектовање квазицикличног кода обично почиње од макроскопског описа структуре кода којим се дефинишу најважнији параметри, као што су тежине врста и колона и кодни количник. Таква макроскопска структура се описује малим бипартитним графом који се назива основним графом (енгл. *base graph*) [74] или протографом [75], а који дефинише позиције n_b група варијабилних и m_b група контролних чворова у финалном Танеровом графу. Основни граф се може описати и матрицом код које, слично контролној матрици кода, јединице одговарају гранама основног графа. Оваква матрица се назива матрицом основног графа (енгл. *base graph matrix*) или матрицом протографа (енгл. *protograph matrix*). Пример је дат на слици 2.5, где је основна матрица обележена са **B**. Танеров граф кода се добија тако



Слика 2.5. Примери конструкције квазицикличног кода из основне матрице и матрице експонената за (а) величину циркуланта $Z = 3$ и (б) величину циркуланта $Z = 32$. Тачке представљају позиције јединица у контролној матрици.

што се основни граф ископира Z пута, чиме се добија m_b група од по Z контролних и k_b група од по Z варијабилних чворова. Затим се гране које повезују чворове пермутују тако да се копије основног графа међусобно повежу. Дозвољене су пермутације само у оквиру група и искључиво су ограничене на кружне пермутације [32]. На пример, варијабилни чворови $v_{i,j}$ су оригинално повезани са контролним чворовима $c_{k,l}$, где су $j,l \in \{0,\ 1,\ 2,\ \ldots,\ Z-1\}$, под условом да су чворови v_i и c_k били повезани у основном графу. Пермутације морају бити такве да контролни чворови k-те групе са индексима $0,\ 1,\ 2,\ \ldots,\ Z-1$ буду повезани са варијабилним чворовима i-те групе са индексима $p,\ p+1,\ \ldots,\ Z-1,\ 0,\ 1,\ \ldots,\ p-1$ респективно, где p представља вредност кружног помераја. Вредности свих кружних помераја могу се сместити у матрицу $\mathbf{P} = \left[\, p_{i,k} \,\right]$ димензија $m_\mathrm{b} \times n_\mathrm{b}$. Тако формирана матрица назива се пермутационом матрицом [38] или матрицом експонената (енгл. *exponent matrix*) [76]. Примери конструкције контролне матрице на основу матрице експонената, за две различите вредности величине циркуланта, приказани су на слици 2.5.

С обзиром на то да је контролна матрица квазицикличног кода јасно структурирана, хардверске реализације и кодера и декодера су значајно ефикасније него код случајних и псеудослучајних кодова. Контролна матрица се може чувати у компримованој форми, тј. довољно је чувати само матрицу експонената, што у значајној мери штеди меморијске ресурсе. Ипак, мана квазицикличних кодова је да се не може увек остварити произвољно мала вероватноћа грешке због појаве такозваних *trapping set*-ова. *Trapping set*-ови представљају структуре у Танеровом графу, чија је последица да се за одређене улазе не може исправити неколико грешака и након много итерација декодовања. Најчешће се манифестују кроз осцилаторно понашање при коме се исправљањем неких заосталих грешака изазову грешке на раније исправљеним битима [65].

2.2.4. LDPC КОДОВИ СА АКУМУЛАЦИЈОМ БИТА ПАРНОСТИ

Кодови са акумулацијом бита парности (енгл. *Repeat-Accumulate, RA*) су кодови код којих се кодна реч формира прогресивном акумулацијом информационих бита у $GF(2)$. Основна идеја је представљена у [77] где је предложено да се кодна реч формира тако што се информациона секвенца дужине k понови q пута, а затим се добијена секвенца пермутује и на крају кодује рекурзивним конволуционим кодером, чија је преносна функција $1/(1+D)$. Пермутација се формално може записати као множење са пермутационом матрицом $\mathbf{\Pi}_{kq\times kq}$, док последњи корак заправо представља акумулацију бита парности јер од секвенце $[a_1\ a_2\ \cdots\ a_n]$, даје излазну секвенцу $[b_1\ b_2\ \cdots\ b_n]$ за коју важи

$$\begin{aligned} b_1 &= a_1, \\ b_2 &= b_1 + a_2 = a_1 + a_2, \\ b_3 &= b_2 + a_3 = a_1 + a_2 + a_3, \\ &\vdots \\ b_n &= b_{n-1} + a_n = a_1 + a_2 + \ldots + a_n. \end{aligned} \tag{10}$$

Добијени код није систематски јер се из кодне речи не може издвојити информациона секвенца, а кодни количник је исти као код кода са понављањем који је примењен у првом кораку $R = 1/q$. Међутим, ако се пре секвенце добијене акумулацијом убаци информациона секвенца, добија се систематски код кодног количника $R = 1/(q+1)$. И овај кôд је заправо

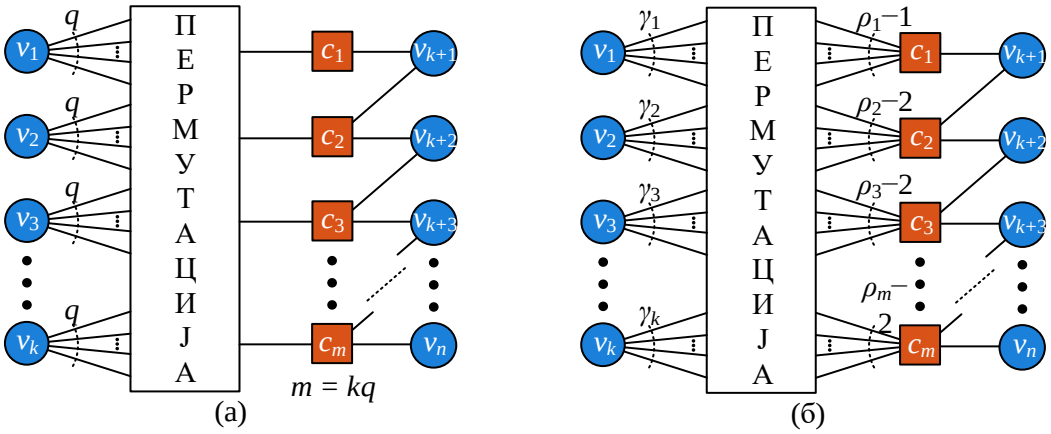
блок кôд за кога се може нацртати Танеров граф. У њему сви информациони варијабилни чворови имају тежину q, а сви осим једног варијабилног чвора који одговарају битима парности имају тежину два. Сви контролни чворови имају тежину три, осим првог који има тежину два. Структура Танеровог графа RA кода приказана је на слици 2.6.а).

Генерализацијом описаног принципа може се добити нерегуларан код било ког кодног количника. Генерална структура Танеровог графа приказана је на слици 2.6.б). Контролни чворови могу бити повезани са више информационих варијабилних чворова, као код било ког LDPC кода. Тако добијен код назива се нерегуларним LDPC кодом са акумулацијом бита парности (енгл. *Irregular Repeat-Accumulate, IRA, LDPC codes*) [78]. Формално записано, контролна матрица IRA LDPC кодова се састоји од две подматрице

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_1 & \mathbf{H}_2 \end{bmatrix}_{m \times n}, \tag{11}$$

од којих је прва случајна матрица, а друга је квадратна двоструко дијагонална матрица:

$$\mathbf{H}_2 = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 1 \end{bmatrix}_{m \times m}. \tag{12}$$



Слика 2.6. Структура Танеровог графа (а) RA кода и (б) IRA кода

Код може бити несистематски, али су значајнији систематски кодови код којих први део кодне речи садржи секвенцу неизмењених информационих бита, а други део кодне речи садржи секвенцу контролних бита. С обзиром на природу контролне матрице, контролни бити се израчунавају акумулацијом секвенце $\begin{bmatrix} \lambda_1 & \lambda_2 & \cdots & \lambda_m \end{bmatrix}$, која се добија множењем матрице $\mathbf{H}_1$ и информационе секвенце $\mathbf{i}$:

$$\boldsymbol{\lambda} = \mathbf{H}_1 \cdot \mathbf{i}. \tag{13}$$

Акумулација се врши као у (10). Оваква структура кода омогућава да се контролна матрица користи и за кодовање, а с обзиром на то да је она ретка, рачунска комплексност кодовања је линеарна. Додатно, иако је у општем случају матрица $\mathbf{H}_1$ случајна, кодовање и декодовање се може додатно поједноставити ако се она пројектује тако да има форму контролне матрице квазицикличних кодова.

2.2.5. Други методи конструкције LDPC кодова

Важни алгоритми за пројектовање LDPC кодова који нису обрађени у претходним одељцима, а могу бити значајни за праксу, приказани су у оквиру овог одељка. Први алгоритам се заснива на постепеном додавању грана Танеровог графа, оптимизованом тако да додавање нове гране омогућава највећу могућу дужину циклуса у локалној околини. Танеров граф кода се формира грану по грану, тј. прогресивно, па је стога на енглеском језику алгоритам назван *Progressive Edge-Growth, PEG* [79]. На тај начин се постиже да цео граф нема кратких циклуса, што даје изузетне перформансе контроле грешака у свим регионима вероватноће грешке. Алгоритам креће од варијабилних чворова и за сваки варијабилни чвор конструише скуп потенцијалних контролних чворова са којима га може повезати, држећи се ограничења да се не направе кратки циклуси. Из наведеног скупа бира се одређени број чворова који тренутно имају најмању тежину, а од њих се онај са којим ће се повезати посматрани варијабилни чвор бира насумично. Тако добијени код има релативно униформну расподелу тежина врста и колона. Значај овог алгоритма је у могућности за аналитичко одређивање доње границе за минимално Хемингово растојање и за дужину минималног циклуса у графу. Међутим, кодови добијени на овај начин нису квазициклични, па је, упркос добрим перформансама контроле грешака, хардверска реализација кодера и декодера комплексна [65], [69].

Поред наведеног алгоритма, треба поменути и алгоритам конструкције кода заснован на латинским квадратима (енгл. *Latin Squares, LS*) [80]. Овај алгоритам омогућава конструкцију структурираног кода са свим структурним особинама квазицикличног кода, код којих се не појављују раније поменуте *trapping set* структуре. Такви кодови имају одличне перформансе контроле грешака у свим нивоима вероватноће грешке и омогућавају ефикасне хардверске реализације кодера и декодера.

2.2.6. LDPC кодови у неким важним комуникационим стандардима

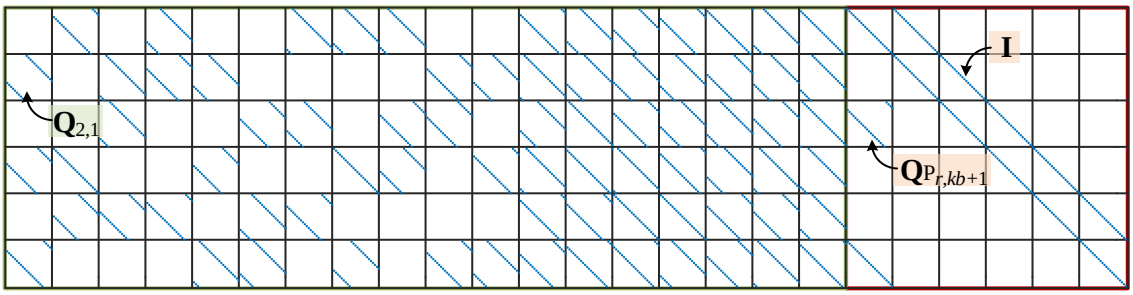
У овом одељку су представљени кодови из неколико комуникационих стандарда који су важни за разумевање каснијих делова текста, а који су коришћени за евалуацију хардверских компоненти пројектованих током рада на докторској дисертацији. С обзиром на то да се користе у практичним системима, сви кодови поменути у овом одељку су структурирани. Кодови из WiMAX, Wi-Fi и 5G NR стандарда су квазициклични док су кодови из DVB-S2 и DVB-S2x стандарда структурирани IRA кодови.

2.2.6.1 WiMAX и Wi-Fi

Телекомуникациони стандард IEEE 802.16e (WiMAX) [28] предвиђа употребу квазицикличних LDPC кодова за кодовање канала. Користе се четири различита кодна количника (1/2, 2/3, 3/4 и 5/6) од којих за два постоје по две различите основне матрице, па се ти кодни количници обележавају са 2/3A, 2/3B, 3/4A и 3/4B. Величина циркуланта Z зависи од дужине информационог блока који се шаље и може узети вредности између 24 и 96 у корацима по четири, док је дужина кодне речи увек $n = 24Z$. Кодови су систематски, а контролна матрица се састоји од две карактеристичне подматрице и у општем облику може се записати у следећој форми:

$$\mathbf{H}=\left[\mathbf{H}_1 \mid \mathbf{H}_2\right]=\left[\begin{array}{cccc|c|ccccccc} \mathbf{Q}_{1,1} & \mathbf{Q}_{1,2} & \cdots & \mathbf{Q}_{1,k_b} & \mathbf{Q}^{P}_{1,k_b+1} & \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{Q}_{2,1} & \mathbf{Q}_{2,2} & \cdots & \mathbf{Q}_{2,k_b} & \mathbf{0} & \mathbf{I} & \mathbf{I} & \cdots & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{Q}_{r-1,1} & \mathbf{Q}_{r-1,2} & \cdots & \mathbf{Q}_{r-1,k_b} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{Q}_{r,1} & \mathbf{Q}_{r,2} & \cdots & \mathbf{Q}_{r,k_b} & \mathbf{Q}^{P}_{r,k_b+1} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I} & \mathbf{I} & \cdots & \mathbf{0} \\ \mathbf{Q}_{r+1,1} & \mathbf{Q}_{r+1,2} & \cdots & \mathbf{Q}_{r+1,k_b} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{I} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{Q}_{m_b,1} & \mathbf{Q}_{m_b,2} & \cdots & \mathbf{Q}_{m_b,k_b} & \mathbf{Q}^{P}_{m_b,k_b+1} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I} \end{array}\right], \qquad (14)$$

где је $\mathbf{I}$ јединична матрица, $\mathbf{0}$ нула-матрица, а $\mathbf{Q}_{i,j}$ циркулант који може бити кружно померена јединична матрица или нула-матрица. Циркуланти који се налазе у матрици $\mathbf{H}_2$ а посебно су обележени са $\mathbf{Q}^{P}_{i,k_b+1}$, описују колоне које одговарају битима парности. Пример једне контролне матрице је приказан на слици 2.7.



Слика 2.7. Контролна матрица WiMAX кода (576, 432) верзија *B*

Кодови из IEEE 802.11 (Wi-Fi) [27] стандарда су исте опште структуре као у (14). Једина разлика је у томе што је матрица $\mathbf{Q}^{P}_{r,k_b+1}$ јединична матрица $\mathbf{I}$, а матрице $\mathbf{Q}^{P}_{1,k_b+1}$ и $\mathbf{Q}^{P}_{m_b,k_b+1}$ су јединичне матрице кружно померене за једну позицију. Овом модификацијом је омогућено за нијансу лакше кодовање када се имплементира кодер који подржава све кодове из стандарда што је детаљније описано у потпоглављу 2.3. Подржани су исти кодни количници као у WiMAX стандарду (1/2, 2/3, 3/4 и 5/6), дужина кодне речи је такође $n=24Z$, а величина циркуланта може узети једну од следеће три вредности: 27, 54 или 81.

2.2.6.2 DVB-S2 и DVB-S2x

Фамилија стандарда за дигитални видео пренос DVB (Digital Video Broadcasting) друге генерације користи структуриране IRA LDPC кодове за заштитно кодовање. DVB стандарди покривају кабловски (DVB-C2), бежични земаљски (DVB-T2) и сателитски видео пренос (DVB-S2 и његова екстензија DVB-S2x) [29]. Сви наведени стандарди користе кодове чије су контролне матрице конструисане на потпуно исти начин, што је погодно за уређаје који треба да подрже сва три стандарда, јер је довољно користити само један заједнички декодер.

Кодне речи могу бити дуге 64800 бита (дугачке речи) или 16200 бита (кратке речи), а у DVB-S2x стандарду постоје и речи средње дужине од 32400 бита. Контролне матрице се, као и код свих IRA кодова, састоје од две подматрице као у (11) и (12). Прва подматрица $\mathbf{H}_1=\left[h_{i,j}\right]$ формира се према следећем правилу:

$$h_{i,j} = \begin{cases} 1, & \text{за } j = \left(x + q(i \bmod Z) \bmod m\right) \\ 0, & \text{иначе} \end{cases}, \tag{15}$$

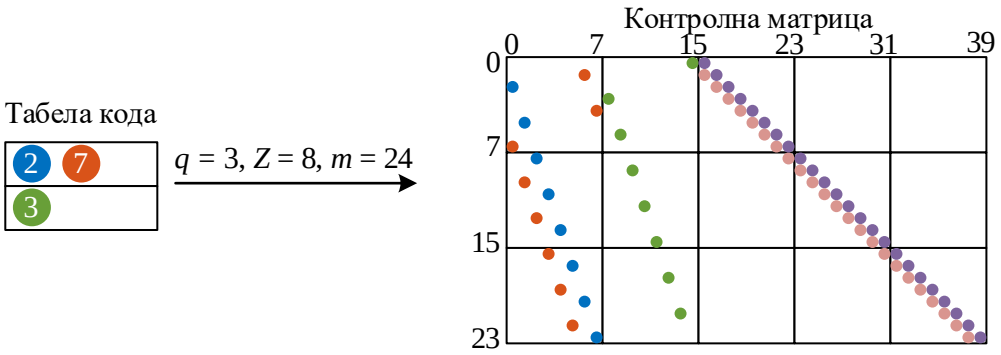
где је m број бита парности, тј. прва димензија контролне матрице, Z константа чија је вредност 360, q има вредност m/Z, x представља број задат у стандардом дефинисаним табелама, док „mod" означава оператор конгруенције. Свака врста табеле описује 360 (Z) колона контролне матрице. Прва врста описује првих 360, друга врста описује других 360 итд. Наведени општи принцип конструкције матрице најлакше је илустровати на примеру. Пример је преузет из [81], а приказан је на слици 2.8 и, ради лакшег разумевања, одабрани параметри кода одговарају знатно мањој контролној матрици него што је предвиђено стандардом. Од једног броја x из табеле генерише се Z јединица, по једна у свакој од Z колона. Вредност x одређује позицију јединице у првој од Z посматраних колона. У свакој следећој колони позиција се одређује додавањем параметра q на позицију из претходне колоне, што је и дефинисано изразом (15). Када се заврши са мапирањем табеле у контролну матрицу, остаје још да се дода двострука дијагонална квадратна матрица, која је карактеристична за све IRA кодове.

Контролне матрице из овог стандарда немају квазицикличну форму. Ипак, како би се омогућила једноставнија хардверска реализација декодера, могуће је одређеним пермутацијама врста и колона постићи приближну квазицикличну форму [81]. Потребно је пермутовати све врсте контролне матрице према следећем правилу:

$$h_{\mathrm{p}_{r_\mathrm{p},v}} = h_{r,v},\ v \in \{0,1,\ldots,n-1\},\ r_\mathrm{p} \in \{0,1,\ldots,m-1\}, \tag{16}$$

где је са $h_{\mathrm{p}_{r_\mathrm{p},v}}$ означен сваки елемент пермутоване матрице, а са $h_{r,v}$ елемент оригиналне матрице, при чему важи

$$r = p\left(r_\mathrm{p} = j + i \cdot Z\right) = \left(i + j \cdot q\right) \bmod m. \tag{17}$$



Слика 2.8. Принцип конструкције кода из DVB фамилије стандарда [81]

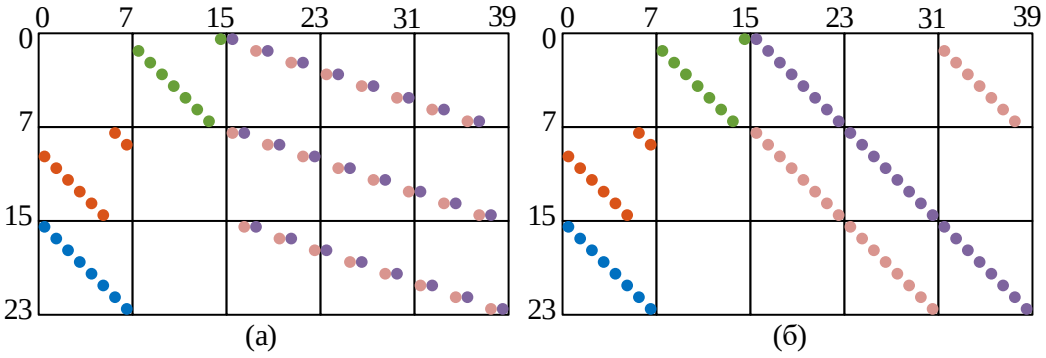
Резултат ове пермутације на примеру са слике 2.8 приказан је на слици 2.9.а). Са слике се види да леви део матрице сада има квазицикличну форму. На крају, остаје да се пермутују колоне у делу матрице који одговара битима парности. Ова пермутација се врши по истом принципу, тј.

$$h_{\mathrm{p}_{r,v_\mathrm{p}}} = h_{r,v},\ v_\mathrm{p} \in \{m, m+1,\ldots,n-1\},\ r \in \{0,1,\ldots,m-1\}, \tag{18}$$

при чему важи

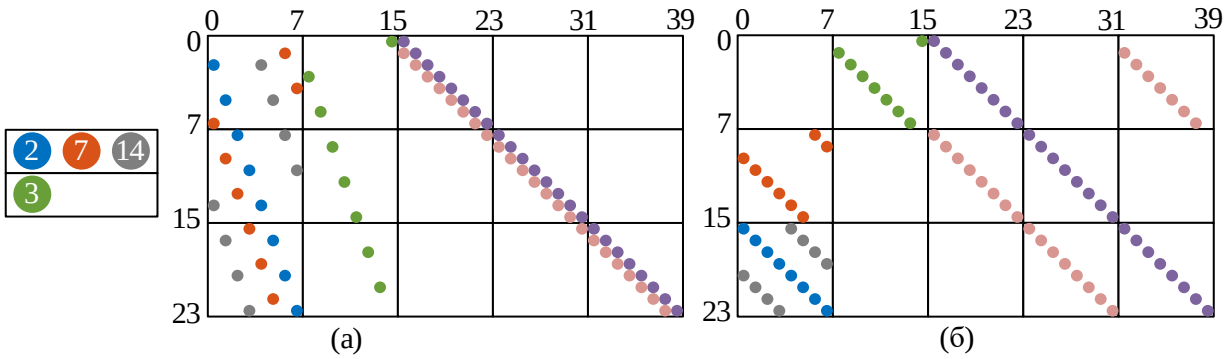
$$v = p\left(v_\mathrm{p} = j + i \cdot Z\right) = \left(i + j \cdot q\right) \bmod m. \tag{19}$$

Финални резултат пермутација матрице са слике 2.8 приказан је на слици 2.9.б). Осим подматрице на горњој десној позицији, која није потпуна кружно померена јединична матрица, остатак контролне матрице има квазицикличну форму. Проблем непотпуног циркуланта посебно се решава у реализацији декодера.



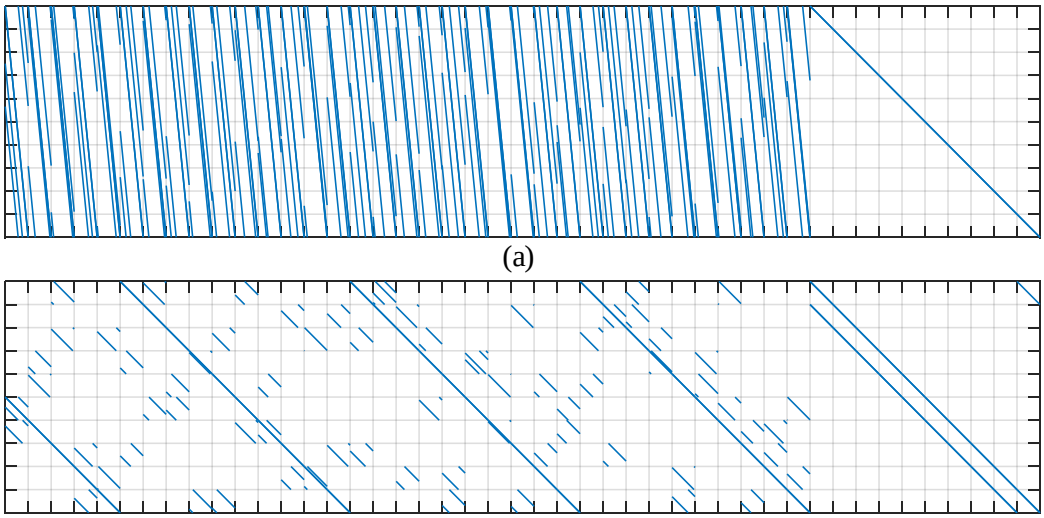
Слика 2.9. Изглед контролне матрице са слике 2.8 након (а) пермутације врста и (б) након пермутације колона које одговарају контролним битима [81]

Иако се контролна матрица DVB кодова може пермутацијама свести на приближну квазицикличну форму, треба нагласити да постоји још један проблем којим се кодови из ових стандарда додатно удаљавају од квазицикличних. Овај проблем настаје у ситуацијама када се у једној врсти табеле, којом се дефинише контролна матрица кода, појаве две вредности чија је разлика целобројни умножак параметра q. Тада се у пермутованој матрици јавља двоструки циркулант, тј. подматрица која се добија збиром две кружно померене јединичне матрице. Поменута ситуација је илустрована на слици 2.10. За неке кодове је могуће да се појаве и вишеструки циркуланти сачињени од три или чак четири кружно померене јединичне матрице. Овај проблем представља посебан изазов у хардверским реализацијама декодера и захтева архитектуралне измене [82] или посебна препроцесирања контролних матрица [81]. Треба напоменути да екстензија сателитског DVB стандарда, DVB-S2x, предвиђа коришћење великог броја кодова који немају проблем са двоструким циркулантима. Притом, наведени кодови дају изузетно добре перформансе контроле грешака. За одређене спектралне ефикасности (информационе протоке нормализоване на предефинисаном пропусном опсегу) DVB-S2x кодови омогућавају вредности SNR-а мање од 1 dB од ултимативне Шенонове границе, што је изузетан резултат за практичан систем [7]. Међутим, с обзиром на то да је стандардом захтевана мала вероватноћа грешке по кодној речи (испод 10^{-5}), поред LDPC кода, као спољашњи кôд, примењен је BCH (*Bose–Chaudhuri–Hocquenghem*) код, који служи за исправљање малог броја грешака заосталих након декодовања LDPC декодером [7].



Слика 2.10. Илустрација појаве двоструког циркуланта у контролној матрици кода из DVB фамилије стандарда: (а) оригинална табела и матрица и (б) матрица након пермутација

На слици 2.11 приказана је контролна матрица кратког кода из DVB-S2 стандарда, кодног количника 7/9, и њена верзија погодна за хардверску реализацију декодера.



(а)

(б)

Слика 2.11. Контролна матрица (16200, 12600) кода из DVB-S2 стандарда: (а) оригинална матрица и (б) матрица након извршених пермутација

2.2.6.3 5G new radio

Пета генерација стандарда за мобилне комуникације 5G New Radio (NR) [31] такође предвиђа коришћење LDPC кодова за заштитно кодовање канала за податке. У ранијим стандардима за мобилне комуникације, UMTS (3G) и LTE (4G), коришћени су турбо кодови, али су корективне способности LDPC кодова и могућност за ефикасније хардверске реализације декодера истиснули турбо кодове из кандидатуре за најновији стандард [32]. Стандард 5G NR подржава велики опсег дужина кодних речи, од неколико стотина до неколико десетина хиљада бита. Подржано је много различитих кодних количника како би се, заједно са разноврсним облицима модулације, обезбедио поуздан пренос информација на великом опсегу вредности односа сигнал-шум. Из тих разлога је број предвиђених LDPC кодова изузетно велики.

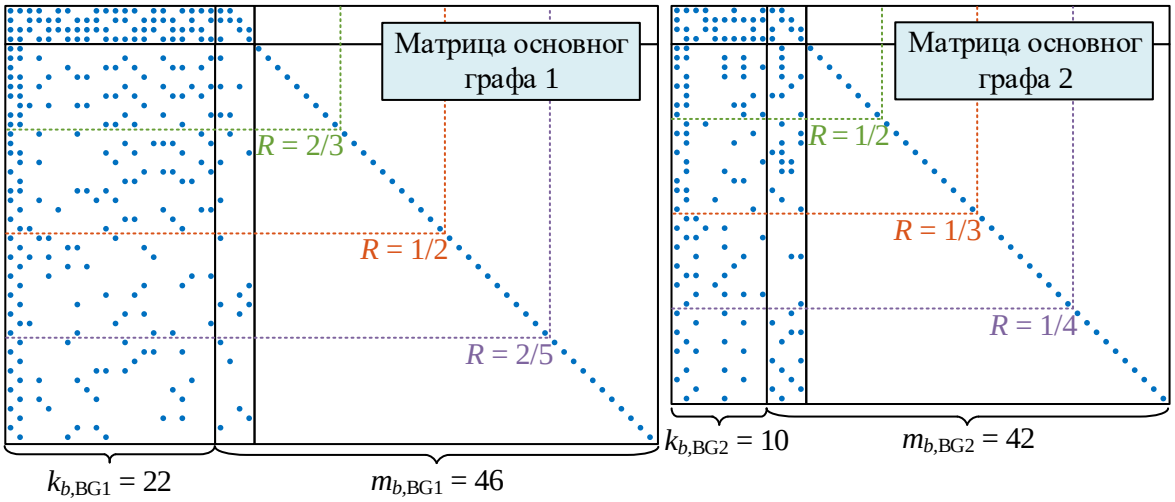
Сви кодови из 5G NR стандарда су квазициклични. Као и код сваког квазицикличног LDPC кода, контролна матрица се састоји од циркуланата величине $Z \times Z$. Кодови су изведени из једног од два основна графа (протографа) који су једноставно названи основни граф 1 (енгл. *base graph* 1, *BG*1) и основни граф 2 (енгл. *base graph* 2, *BG*2). Општа структура контролне матрице може se записати са

$$\mathbf{H} = \begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{0} \\ \mathbf{C} & \mathbf{D} & \mathbf{I} \end{bmatrix}, \tag{20}$$

где матрице **A** (димензија $4Z \times k_b Z$), **C** (димензија $(m_b - 4)Z \times k_b Z$), и **D** (димензија $(m_b - 4)Z \times 4Z$), имају општу квазицикличну форму, док матрица **B** (димензија $4Z \times 4Z$) има структуру сличну структури десне подматрице контролне матрице кода из Wi-Fi стандарда. У зависности од протографа и величине циркуланата, матрица **B** може имати једну од следеће четири структуре:

$$\mathbf{B}_{\mathrm{BG1,1}}=\begin{bmatrix}\mathbf{I}^{(1)} & \mathbf{I} & \mathbf{0} & \mathbf{0}\\ \mathbf{I} & \mathbf{I} & \mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{I}\\ \mathbf{I}^{(1)} & \mathbf{0} & \mathbf{0} & \mathbf{I}\end{bmatrix},\ \mathbf{B}_{\mathrm{BG1,2}}=\begin{bmatrix}\mathbf{I} & \mathbf{I} & \mathbf{0} & \mathbf{0}\\ \mathbf{I}^{(105\bmod Z)} & \mathbf{I} & \mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{I}\\ \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{I}\end{bmatrix},$$
$$\mathbf{B}_{\mathrm{BG2,1}}=\begin{bmatrix}\mathbf{I} & \mathbf{I} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I} & \mathbf{I} & \mathbf{0}\\ \mathbf{I}^{(1)} & \mathbf{0} & \mathbf{I} & \mathbf{I}\\ \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{I}\end{bmatrix},\ \mathbf{B}_{\mathrm{BG2,2}}=\begin{bmatrix}\mathbf{I}^{(1)} & \mathbf{I} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I} & \mathbf{I} & \mathbf{0}\\ \mathbf{I} & \mathbf{0} & \mathbf{I} & \mathbf{I}\\ \mathbf{I}^{(1)} & \mathbf{0} & \mathbf{0} & \mathbf{I}\end{bmatrix}, \tag{21}$$

где је са $\mathbf{I}^{(p)}$ означена јединична матрица кружно померена за p позиција. Резултат операције $105 \bmod Z$ може бити само 105 или 1. Матрице оба основна графа приказане су на слици 2.12. Основни кодни количници су $R_{\mathrm{BG1}}=22/68$ и $R_{\mathrm{BG2}}=22/52$, док се виши кодни количници добијају скраћењем контролне матрице (енгл. *shortening*). Одбацивањем одређеног броја последњих врста и истог броја последњих колона могу се добити кодни количници до максимално $22/27$ за граф 1 и $10/15$ за граф 2.



Слика 2.12. Структура контролних матрица кодова из 5G NR стандарда

Кодови су систематски, па кодне речи имају следећу форму:

$$\mathbf{x}=\begin{bmatrix}\mathbf{i} & \mathbf{p}_{\mathrm{c}} & \mathbf{p}_{\mathrm{a}}\end{bmatrix}, \tag{22}$$

где је са $\mathbf{i}$ означена секвенца информационих бита дужине $k_{\mathrm{b}}Z$, док је са $\mathbf{p}_{\mathrm{c}}$ означена секвенца такозваних основних контролних бита (енгл. *core parity bits*) дужине $4Z$, а са $\mathbf{p}_{\mathrm{a}}$ секвенца додатних контролних бита (енгл. *extended parity bits*) дужине $(m_{\mathrm{b}}-4)Z$. Дужина кодне речи је стога $n_{\mathrm{b}}Z$ где је $n_{\mathrm{b}}=k_{\mathrm{b}}+m_{\mathrm{b}}$ [32]. Треба поменути да се првих $2Z$ информационих бита никада не шаље у канал, чиме се ефективни кодни количник повећава на $R=k_{\mathrm{b}}/(n_{\mathrm{b}}-2)$. Ова техника се назива пунктирање („бушење") (енгл. *puncturing*). Због пунктирања, прве две колоне основне матрице имају значајно већу тежину од осталих, што је урађено да би се у декодеру лакше реконструисали информациони бити који нису послати, али и због тога што је показано да колоне велике тежине доприносе бољим перформансама контроле грешака [32]. Поред пунктирања, ефективни кодни количник се мења и због тога што је дозвољено да информациони садржај буде произвољне дужине. За величину

циркуланта се бира најмања вредност која омогућава да је $k_b Z$ веће или једнако од броја информационих бита. У ситуацијама када је број информационих бита мањи од $k_b Z$, информациона секвенца се допуњава нулама при кодовању, али се нуле не шаљу кроз канал. На страни декодера се информација да су неки бити били нуле може искористити за ефикасније декодовање [83].

Величина циркуланта може имати било коју вредност између 2 и 384 која се може записати у форми

$$Z = a \cdot 2^j, \tag{23}$$

где a може узети вредности из скупа $\{2, 3, 5, 7, 9, 11, 13, 15\}$. Оваква дефиниција даје укупно 51 могућу вредност величине циркуланта што, са свим могућим скраћењима матрица, даје преко 4000 различитих кодова.

Матрицама основног графа придружена је по једна пермутациона матрица за сваку вредност параметра a из (23) у којој су вредности кружног помераја дефинисане за максималну величину циркуланта (256, 384, 320, 224, 288, 352, 208 или 240). Све друге пермутационе матрице су истих димензија и имају следећи облик:

$$\mathbf{P} = \begin{bmatrix} P_{1,1} & P_{1,2} & \cdots & P_{1,n_b} \\ P_{2,1} & P_{2,2} & \cdots & P_{1,n_b} \\ \vdots & \vdots & \ddots & \vdots \\ P_{m_b,1} & P_{m_b,2} & \cdots & P_{m_b,n_b} \end{bmatrix}, \tag{24}$$

а вредности $P_{i,j}$ добијају се на следећи начин:

$$P_{i,j} = \begin{cases} -1, & \text{ако је } V_{i,j} = -1 \\ V_{i,j} \bmod Z, & \text{ако је } V_{i,j} > -1 \end{cases}, \tag{25}$$

где су са $V_{i,j}$ обележене вредности из оригиналне пермутационе матрице, а са Z жељена величина циркуланта [31].

Постоји низ, стандардом дефинисаних, правила којима се одређује који се основни граф користи и који кодни количник. Дата је подршка за додатно смањење кодног количника при изузетно лошим условима преноса понављањем одређеног броја бита из кодне секвенце. У случајевима када је декодовање неуспешно користе се HARQ процедуре са инкременталном редундансом. О овим и другим детаљима више се може прочитати у стандарду [31], као и у радовима [32] и [76].

2.3. Кодовање LDPC кодова

У зависности од структуре и особина LDPC кода, кодовање се може урадити на више различитих начина. У овом потпоглављу описани су најчешће коришћени методи примењени у кодерима LDPC кодова. Неки основни концепти су већ представљени раније, али ће овде бити систематизовани са циљем разумевања њихових предности и мана.

2.3.1. Директно кодовање

Директно кодовање представља множење информационе секвенце генеришућом матрицом, тј. примену једначине (1). С обзиром на то да се LDPC кодови пројектују

конструкцијом контролне матрице, генеришућа матрица се најчешће добија из (4) коришћењем Гаусове елиминације.

Иако принципски најједноставнији, овај метод је рачунски најкомплекснији. Основни разлог за то је густина генеришуће матрице. За разлику од контролне матрице која се пројектује да буде ретка, генеришућа матрица најчешће није ретка, па комплексност кодовања расте квадратно са дужином кодне речи. Друга мана овог метода је што готово да не може бити флексибилан. За сваки код је неопходно израчунати (и чувати!) генеришућу матрицу, што је неприхватљиво за примену у флексибилним стандардима какви су они описани у одељку 2.2.6. Ипак, сви кодови из поменутих стандарда имају структуру која омогућава ефикасније кодовање о чему говоре наредни одељци.

Међутим, за одређене структуре матрице и када флексибилност није императив, директно кодовање се може применити о чему сведочи неколико ефикасних хардверских реализација кодера које суштински користе овај метод [84]–[88].

2.3.2. ДВОСТЕПЕНО КОДОВАЊЕ

Двостепено кодовање (енгл. *partitioned parity-check matrix two-step encoding*) може се применити за кодовање систематских кодова. У том случају се контролна матрица, макар она била и случајна, може поделити на две подматрице $\mathbf{H}=\begin{bmatrix}\mathbf{H}_1 & \mathbf{H}_2\end{bmatrix}$. За сваки систематски код тада важи следећа једнакост:

$$\begin{bmatrix}\mathbf{H}_1 & \mathbf{H}_2\end{bmatrix}\begin{bmatrix}\mathbf{i} & \mathbf{p}\end{bmatrix}^{\mathrm{T}}=\mathbf{H}_1\mathbf{i}^{\mathrm{T}}+\mathbf{H}_2\mathbf{p}^{\mathrm{T}}=\mathbf{0}. \tag{26}$$

Из наведене једнакости се лако може извести израз за вектор контролних бита:

$$\mathbf{p}^{\mathrm{T}}=\mathbf{H}_2^{-1}\mathbf{H}_1\mathbf{i}^{\mathrm{T}}. \tag{27}$$

На овај начин се постиже смањење комплексности кодовања јер је матрица $\mathbf{H}_1$ ретка, па је комплексност израчунавања вектора $\mathbf{H}_1\mathbf{i}^{\mathrm{T}}$ линеарна. Једини услов је да је матрица $\mathbf{H}_2$ несингуларна у $GF(2)$. Међутим, комплексност израчунавања је и даље велика јер матрица $\mathbf{H}_2^{-1}$ није ретка, што доводи до великог заузећа хардверских ресурса. Закључци везани за флексибилност и чување матрице који важе за директно кодовање, важе и за овај метод.

Хардверске реализације које користе овај метод приказане су у радовима [89]–[91]. Иако сам метод има велику рачунску комплексност, у овим радовима су приказане ефикасне хардверске реализације кодера које поједностављују комплексност множења густом матрицом $\mathbf{H}_2^{-1}$ вишеструким коришћењем међурезултата, а које је могуће захваљујући погодној структури кода.

2.3.3. МЕТОД РИЧАРДСОНА И УРБАНКЕА

Ричардсон и Урбанке [92] су предложили метод за ефикасно кодовање LDPC кодова којим се може остварити приближно линеарна рачунска комплексност за велики број кодова. Метод се састоји од два корака: 1) препроцесирања контролне матрице и 2) ефикасног кодовања. У првом кораку се пермутују врсте и колоне контролне матрице тако да пермутована матрица има приближну доње-троугаону форму, тј. форму облика

$$\mathbf{H}=\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{T}\\ \mathbf{C} & \mathbf{D} & \mathbf{E}\end{bmatrix}, \tag{28}$$

где су матрице $\mathbf{A}$ (димензија $(m-g)\times k$), $\mathbf{B}$ (димензија $(m-g)\times g$), $\mathbf{C}$ (димензија $g\times k$), $\mathbf{D}$ (димензија $g\times g$) и $\mathbf{E}$ (димензија $g\times(m-g)$) ретке матрице, при чему је g цео број, док је матрица $\mathbf{T}$ ретка квадратна доње-троугаона матрица димензија $(m-g)\times(m-g)$. У том случају се кодна реч може представити са

$$\mathbf{x}=\begin{bmatrix}\mathbf{i} & \mathbf{p}_1 & \mathbf{p}_2\end{bmatrix}, \tag{29}$$

где је са $\mathbf{i}$ означена секвенца информационих бита дужине k, док је са $\mathbf{p}_1$ означена секвенца првог дела контролних бита дужине g, а са $\mathbf{p}_2$ секвенца другог дела контролних бита дужине $m-g$. Пермутације се изводе похлепним алгоритмима са циљем да се обезбеди описана форма контролне матрице уз ограничење да је матрица $-\mathbf{E}\mathbf{T}^{-1}\mathbf{B}+\mathbf{D}$ несингуларна у $GF(2)$. Додатно, тежи се да вредност параметра g буде што је могуће мања, по могућству сразмерна са $\sqrt{n}$.

Ако се контролна матрица успешно сведе на форму (28), онда се контролни бити могу израчунати коришћењем следећих израза:

$$\mathbf{p}_1^{\mathrm{T}}=\left(-\mathbf{E}\mathbf{T}^{-1}\mathbf{B}+\mathbf{D}\right)^{-1}\left(-\mathbf{E}\mathbf{T}^{-1}\mathbf{A}+\mathbf{C}\right)\mathbf{i}^{\mathrm{T}}, \tag{30}$$

$$\mathbf{p}_2^{\mathrm{T}}=-\mathbf{T}^{-1}\left(\mathbf{A}\mathbf{i}^{\mathrm{T}}+\mathbf{B}\mathbf{p}_1^{\mathrm{T}}\right). \tag{31}$$

Множења са $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ и $\mathbf{E}$ су множења ретким матрицама чија је рачунска комплексност линеарна. Сабирања такође имају линеарну рачунску комплексност. С обзиром на то да је $\mathbf{T}$ доње-троугаона матрица, сва множења са $\mathbf{T}^{-1}$, која је у општем случају густа матрица, могу се заменити процесом који ипак има линеарну рачунску комплексност. Ова чињеница је заснована на томе да се вредност израза $\mathbf{b}=\mathbf{T}^{-1}\mathbf{a}$ може наћи решавањем еквивалентног система једначина $\mathbf{T}\mathbf{b}=\mathbf{a}$, а који се ефикасно решава супституцијом унапред (енгл. *forward substitution*), чија је комплексност линеарна када је матрица $\mathbf{T}$ ретка. Једино множење са $\left(-\mathbf{E}\mathbf{T}^{-1}\mathbf{B}+\mathbf{D}\right)^{-1}$ има квадратну комплексност израчунавања. Ова матрица има димензије $g\times g$, па се управо због тога приликом препроцесирања тежи да параметар g буде што је могуће мањи [65], [58]. Приближно линеарна рачунска комплексност процеса кодовања је главни разлог због ког је велики број практичних реализација кодера заснован на баш овом методу [93]–[99].

Мана описаног метода је мали потенцијал за флексибилне реализације, јер је сваку кодну матрицу потребно препроцесирати и чувати резултат густе матрице $\left(-\mathbf{E}\mathbf{T}^{-1}\mathbf{B}+\mathbf{D}\right)^{-1}$ [86], [58]. Поред тога, из (30) и (31) се види да сам процес израчунавања контролних бита има много секвенцијалних корака што потенцијално резултира дугачким критичним путањама у хардверским реализацијама кодера [100].

2.3.4. ХИБРИДНО КОДОВАЊЕ

Са циљем да се избегне множење са претходно поменутом густом матрицом $\left(-\mathbf{E}\mathbf{T}^{-1}\mathbf{B}+\mathbf{D}\right)^{-1}$ из (30), Коен и Пархи [100] су предложили хибридни метод кодовања за кодове из IEEE 802.3an (10GBASE-T) стандарда. Предложено је да се за израчунавање првих g контролних бита ($\mathbf{p}_1$) користи генеришућа матрица уместо једначине (30), док се за израчунавање преосталих контролних бита ($\mathbf{p}_2$) користи метод Ричардсона и Урбанкеа. На

овај начин је омогућена лакша трансформација контролне матрице пермутацијама, јер услов несингуларности матрице $-\mathbf{E}\mathbf{T}^{-1}\mathbf{B}+\mathbf{D}$ више није неопходан. Такође, критична путања је скраћена, што може довести до бржих хардверских реализација [58]. Поред рада [100], овај метод је недавно примењен и за реализацију кодера за кодове у сателитским комуникацијама предложене стандардом комитета *Consultative Committee for Space Data Systems* (*CCSDS*) [101].

2.3.5. КОДОВАЊЕ ЗАСНОВАНО НА СУПСТИТУЦИЈИ

Велики број стандардизованих LDPC кодова има структуру која омогућава ефикасније кодовање него што омогућава и један од претходно описаних метода. Ефикасно кодовање погодно структурираних кодова је засновано на супституцији, тј. на ефикасном решавању система једначина. Овај је метод најједноставније описати на практичном примеру, па су за те потребе у овом одељку коришћени кодови из Wi-Fi стандарда пошто је тај приступ најчешће и коришћен за хардверске реализације Wi-Fi кодера [102]–[105].

Структура свих LDPC кодова из Wi-Fi стандарда је квазициклична, а контролна матрица има следећу форму:

$$\mathbf{H}=\left[\mathbf{H}_1 \mid \mathbf{H}_2\right]=\left[\begin{array}{cccc|cccccccc} \mathbf{Q}_{1,1} & \mathbf{Q}_{1,2} & \cdots & \mathbf{Q}_{1,k_\mathrm{b}} & \mathbf{I}^{(1)} & \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{Q}_{2,1} & \mathbf{Q}_{2,2} & \cdots & \mathbf{Q}_{2,k_\mathrm{b}} & \mathbf{0} & \mathbf{I} & \mathbf{I} & \cdots & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{Q}_{r-1,1} & \mathbf{Q}_{r-1,2} & \cdots & \mathbf{Q}_{r-1,k_\mathrm{b}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{Q}_{r,1} & \mathbf{Q}_{r,2} & \cdots & \mathbf{Q}_{r,k_\mathrm{b}} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I} & \mathbf{I} & \cdots & \mathbf{0} \\ \mathbf{Q}_{r+1,1} & \mathbf{Q}_{r+1,2} & \cdots & \mathbf{Q}_{r+1,k_\mathrm{b}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{I} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{Q}_{m_\mathrm{b},1} & \mathbf{Q}_{m_\mathrm{b},2} & \cdots & \mathbf{Q}_{m_\mathrm{b},k_\mathrm{b}} & \mathbf{I}^{(1)} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I} \end{array}\right]. \tag{32}$$

Ако се информациони и контролни бити групишу у групе од по Z бита, кодна реч се може представити у следећој форми:

$$\mathbf{x}=\left[\mathbf{i}_1 \quad \mathbf{i}_2 \quad \cdots \quad \mathbf{i}_{k_\mathrm{b}} \quad \mathbf{p}_1 \quad \mathbf{p}_2 \quad \cdots \quad \mathbf{p}_{m_\mathrm{b}}\right]. \tag{33}$$

С обзиром на то да важи $\mathbf{H}\cdot\mathbf{x}^{\mathrm{T}}=\mathbf{0}$, из (32) и (33) може се извести следећи систем једначина у $GF(2)$:

$$\begin{aligned} &\sum_{l=1}^{k_\mathrm{b}}\mathbf{Q}_{1,l}\mathbf{i}_l^{\mathrm{T}}+\mathbf{p}_1^{\mathrm{T}(1)}+\mathbf{p}_2^{\mathrm{T}}=\mathbf{0}, \text{ (први ред)},\\ &\sum_{l=1}^{k_\mathrm{b}}\mathbf{Q}_{j,l}\mathbf{i}_l^{\mathrm{T}}+\mathbf{p}_j^{\mathrm{T}}+\mathbf{p}_{j+1}^{\mathrm{T}}=\mathbf{0}, \text{ за } j\in\{2,3,\ldots,r-1\}\cup\{r+1,r+2,\ldots,m_\mathrm{b}-1\},\\ &\sum_{l=1}^{k_\mathrm{b}}\mathbf{Q}_{r,l}\mathbf{i}_l^{\mathrm{T}}+\mathbf{p}_1^{\mathrm{T}}+\mathbf{p}_r^{\mathrm{T}}+\mathbf{p}_{r+1}^{\mathrm{T}}=\mathbf{0}, \text{ (r-ти ред)},\\ &\sum_{l=1}^{k_\mathrm{b}}\mathbf{Q}_{m_\mathrm{b},l}\mathbf{i}_l^{\mathrm{T}}+\mathbf{p}_1^{\mathrm{T}(1)}+\mathbf{p}_{m_b}^{\mathrm{T}}=\mathbf{0}, \text{ (m_b-ти ред)}. \end{aligned} \tag{34}$$

Сабирањем свих наведених једначина добија се израз за прву групу контролних бита:

$$\mathbf{p}_1^{\mathrm{T}} = \sum_{j=1}^{m_\mathrm{b}} \sum_{l=1}^{k_\mathrm{b}} \mathbf{Q}_{j,l} \mathbf{i}_l^{\mathrm{T}}. \tag{35}$$

Међурезултати у суми (35) могу се обележити са

$$\boldsymbol{\lambda}_j = \sum_{l=1}^{k_\mathrm{b}} \mathbf{Q}_{j,l} \mathbf{i}_l^{\mathrm{T}}. \tag{36}$$

Производ $\mathbf{Q}_{j,l}\mathbf{i}_l^{\mathrm{T}}$ није ништа друго до кружно померена секвенца информационих бита $\mathbf{i}_l$, јер је $\mathbf{Q}_{j,l}$ кружно померена јединична матрица. Наравно, ако је $\mathbf{Q}_{j,l}$ нула-матрица и резултат операције $\mathbf{Q}_{j,l}\mathbf{i}_l^{\mathrm{T}}$ је секвенца нула. Стога се међурезултати $\boldsymbol{\lambda}_j$ у практичним реализацијама кодера добијају сумирањем кружно померених подсеквенци информационих бита. Преостали контролни бити се срећивањем система (34) и (35) добијају коришћењем следећих израза:

$$\begin{gathered}
\mathbf{p}_2^{\mathrm{T}} = \boldsymbol{\lambda}_1 + \mathbf{p}_1^{\mathrm{T}(1)}; \qquad \mathbf{p}_{j+1}^{\mathrm{T}} = \boldsymbol{\lambda}_j + \mathbf{p}_j^{\mathrm{T}}, \text{ за } j \in \{2,3,\ldots,r-2\}; \\
\mathbf{p}_{m_\mathrm{b}}^{\mathrm{T}} = \boldsymbol{\lambda}_{m_\mathrm{b}} + \mathbf{p}_1^{\mathrm{T}(1)}; \quad \mathbf{p}_j^{\mathrm{T}} = \boldsymbol{\lambda}_j + \mathbf{p}_{j+1}^{\mathrm{T}}, \text{ за } j \in \{m_\mathrm{b}-1, m_\mathrm{b}-2,\ldots,r+1\}; \\
\mathbf{p}_r^{\mathrm{T}} = \boldsymbol{\lambda}_r + \mathbf{p}_1^{\mathrm{T}(1)} + \mathbf{p}_{r+1}^{\mathrm{T}}, \text{ ако је } r > 2.
\end{gathered} \tag{37}$$

Веома сличан приступ је недавно примењен на кодовање 5G NR LDPC кодова [106], што је и очекивано с обзиром на то да је велики део контролне матрице 5G NR кодова сличан контролним матрицама Wi-Fi кодова. Тај метод је коришћен и у овој докторској дисертацији и у претходно публикованом раду [58], што је детаљније описано у поглављу 3. Погодну структуру матрице имају и неки други кодови, па се за њих може применити кодовање засновано на супституцији. Када то није могуће, уобичајен приступ је кодовање методом Ричардсона и Урбанкеа.

Важна предност описаног метода у односу на остале је та што се за кодовање користи искључиво контролна матрица, тј. не постоје инверзије матрица, нити рачунски комплексна множења густим матрицама. С обзиром на структуру кодова за које се примењује овакво кодовање, контролна матрица се обично може чувати у компримованој форми као што је описано у одељку 2.2.3. Због свега тога могуће је постићи велику флексибилност и подршку за велики број кодова уз мале додатне хардверске ресурсе. Међутим, очигледно је да и даље постоји одређени број секвенцијалних израчунавања која ограничавају проток кодера. То би било могуће решити искључиво веома паралелним реализацијама које би, с друге стране, смањиле флексибилност.

2.4. Декодовање LDPC кодова

LDPC кодови се најчешће декодују итеративним алгоритмима. Неке од њих је већ предложио Галагер у свом оригиналном раду [21] и у својој докторској дисертацији [107] заједно са описом самих кодова. Алгоритми декодовања се могу поделити на две категорије. Прва категорија алгоритама је заснована на комплементирању бита кодне речи за које већина провера парности није задовољена и у литератури се назива *bit-flipping* алгоритмима. Друга категорија алгоритама је заснована на размени порука између варијабилних и контролних чворова Танеровог графа и у литератури се назива *message-passing, MP*, алгоритмима. Обе врсте алгоритама могу радити са тврдим одлукама (енгл. *hard-decision, HD*), које представљају неке једнобитне информације, или са меким одлукама (енгл. *soft-decision, SD*), које најчешће представљају одређене вероватноће да поменуте информације имају вредност

„0” или „1”. У овом потпоглављу представљени су основни концепти алгоритама декодовања LDPC кодова и њихове разлике у погледу перформанси контроле грешака, комплексности рачунских операција и поља примене.

2.4.1. ДЕКОДОВАЊЕ ЗАСНОВАНО НА ТВРДИМ ОДЛУКАМА

Један од најједноставнијих алгоритама декодовања LDPC кодова заснован је на итеративном мењању процењених бита кодне речи на основу тренутне вредности синдрома. Један бит кодне речи се комплементира ако већина контролних сума у које тај бит улази није задовољена, па се због тога алгоритам и назива *bit-flipping* алгоритмом. Поступак декодовања је итеративан, а једна итерација има следеће кораке.

- На основу тренутне процене кодне речи израчунају се тренутне вредности синдрома, тј. вредности свих једначина провера парности у сваком од контролних чворова: $\mathbf{c} = \mathbf{v} \cdot \mathbf{H}^{\mathrm{T}}$.
- За оне једначине провера парности које нису задовољене, идентификују се сви бити тренутне процене кодне речи који у њима учествују. Ови бити су кандидати за комплементирање.
- Одреди се број незадовољених провера парности у којима учествује сваки од претходно одабраних кандидата.
- Комплементирају се они кандидати за које је број незадовољених провера парности већи од половине укупног броја контролних сума у којима учествују.

Описани поступак се понавља све док синдром не постане једнак нула-вектору, чиме је декодовање успешно извршено, или док се не дође до предефинисаног максималног броја итерација [107].

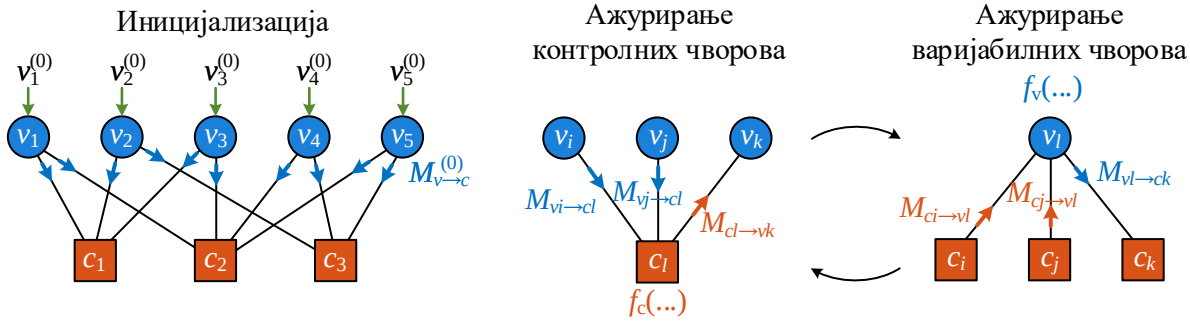
С обзиром на то да се све процене кодне речи комплементирају истовремено, описани алгоритам се назива још и паралелним *bit-flipping* алгоритмом. Декодовање је могуће урадити и ако се бити комплементирају један по један, тј. ако се комплементира онај бит за који највећи број провера парности није задовољен, а затим уради поновно ажурирање кандидата за комплементирање. На тај начин може се десити да бити који учествују у истим контролним сумама као и комплементирани бит, имају знатно већи број задовољених провера парности него раније. Тиме се избегава непотребно комплементирање великог броја бита, али се, честим ажурирањем кандидата за комплементирање, декодер знатно успорава. Описани метод назива се серијским или секвенцијалним *bit-flipping* алгоритмом [108].

Рачунска комплексност *bit-flipping* алгоритма није велика, па су и његове хардверске реализације веома ефикасне. Неке сложеније варијанте овог алгоритма могу дати боље перформансе контроле грешака, али уз повећану рачунску комплексност. Међу њима су значајни *bit-flipping* алгоритам са тежинским фактором (енгл. *weighted bit-flipping*) [71] и модификовани *bit-flipping* алгоритам са тежинским фактором [109], код којих се одлука о инвертовању доноси на основу броја незадовољених провера парности помноженог тежинским фактором који зависи од примљене вредности из канала. Декодовање засновано на оптимизационом поступку градијентним методом предложено је у [110] где је представљен такозвани *gradient descent bit-flipping* алгоритам, док су у [111] коришћене меке поруке којима се у одређеној мери дефинише поузданост да су одређени бити процене кодне речи нуле или јединице, као и поузданост да контролне суме имају неку од ове две вредности.

Декодовање засновано на тврдим одлукама може се реализовати и коришћењем метода размене порука између варијабилних и контролних чворова Танеровог графа. Оригинално, овај метод је представио Галагер у свом, већ више пута поменутом, раду [21] и у својој докторској дисертацији [107]. Најједноставније варијанте овог алгоритма, код кога су поруке

бинарне, познате су под називом Галагер-А и Галагер-Б (енгл. *Gallager-A/B*) и оне су представљене у овом одељку, док су сложенији приступи представљени у одељку 2.4.2.

Општи принцип свих алгоритама са размена порука (енгл. *message-passing, MP*) подразумева следеће кораке: иницијализацију и итеративно ажурирање контролних и варијабилних чворова све док се не достигне максималан број итерација или док све провере парности не буду задовољене. Иницијализацијом се стања варијабилних чворова постављају на вредности одређене сигналима примљеним из канала. Затим варијабилни чворови шаљу поруке контролним чворовима које генеришу на основу својих тренутних стања. Ове поруке су у даљем тексту називане порукама варијабилних чворова и обележаване са $M_{v\to c}$ (од енгл. *variable-to-check messages*). Контролни чворови примају наведене поруке и на основу њих генеришу своје поруке као одговор. Ове поруке су у даљем тексту називане порукама контролних чворова и обележаване са $M_{c\to v}$ (од енгл. *check-to-variable messages*). Порука контролног чвора c (тежине ρ_c), која се шаље варијабилном чвору v, резултат је одређене функције која је имплементирана у том чвору. Сама функција зависи од алгоритма декодовања, али су њени аргументи увек вредности порука које су дошле од $\rho_c - 1$ суседних[1] варијабилних чворова чвору v. Суштински, контролни чворови говоре коју вредност би требало да имају варијабилни чворови да би провера парности била задовољена. Након што варијабилни чворови приме поруке од контролних чворова, као одговор шаљу нове поруке контролним чворовима и ажурирају своја стања, чиме се завршава једна итерација декодовања. Слично контролним чворовима, и варијабилни чворови генеришу поруке које по гранама графа путују ка контролним чворовима, а израчунавају се на основу претходно пристиглих порука њима суседних контролних чворова, као и на основу тренутног стања варијабилних чворова. Функција израчунавања у варијабилним чворовима такође зависи од алгоритма декодовања.



Слика 2.13. Илустрација алгоритма декодовања заснованог на размени порука

Код Галагер-А и Галагер-Б алгоритама иницијализација варијабилних чворова врши се тврдим одлукама, тј. нулама и јединицама генерисаним на основу примљених вредности сигнала из канала. Самим тим су генерисане и све иницијалне поруке варијабилних чворова:

$$M_{v\to c} = y_v, \forall v \in \{1,2,\ldots,n\}, \forall c \in \{1,2,\ldots,m\} \tag{38}$$

Ажурирање контролних чворова у итерацији it и уједно генерисање порука врши се према следећем правилу:

$$M_{c\to v}^{(it)} = \left(\sum_{v' \in V_c \setminus v} M_{v'\to c}^{(it-1)} \right) \bmod 2 \tag{39}$$

[1] Раније је поменуто да су суседни варијабилни чворови они чворови који су повезани на исти контролни чвор. Слично, суседни контролни чворови су они чворови који су повезани на исти варијабилни чвор.

где је са V_c обележен скуп свих варијабилних чворова повезаних на контролни чвор c. У варијабилним чворовима се затим генеришу нове поруке према следећем правилу:

$$M_{v\to c}^{(it)} = \begin{cases} 1, & \text{ако је } \sum_{c'\in C_v\setminus c} M_{c'\to v}^{(it)} \geq b_{\gamma_v}^{(it)} \\ 0, & \text{ако је } \sum_{c'\in C_v\setminus c} M_{c'\to v}^{(it)} \leq \gamma_v - 1 - b_{\gamma_v}^{(it)}, \\ y_v, & \text{иначе} \end{cases} \tag{40}$$

где је са C_v обележен скуп свих контролних чворова повезаних на варијабилни чвор v, са γ_v је обележена тежина тог варијабилног чвора, а са $b_{\gamma_v}^{(it)}$ вредност прага која зависи од γ_v и типа алгоритма. Ако је праг $b_{\gamma_v}^{(it)}$ константан у свим итерацијама и ако је $b_{\gamma_v}^{(it)} = \gamma_v - 1$, тада је реч о Галагер-А алгоритму, а када је $b_{\gamma_v}^{(it)} = \lceil \gamma_v / 2 \rceil$, реч је о Галагер-Б алгоритму. Након генерисања порука, остаје да се у текућој итерацији одреде стања варијабилних чворова, тј. да се процене вредности бита кодне речи. Оне се одређују већинском логиком на основу иницијалних вредности и свих примљених порука контролних чворова.

Алгоритми који користе тврде одлуке немају много примена у бежичним комуникацијама, с обзиром на то да немају добре перформансе у каналу са белим Гаусовим шумом (енгл. *Additive White Gaussian Noise, AWGN, channel*). Међутим, у бинарном симетричном каналу (енгл. *Binary Symmetric Channel, BSC*), или бинарном каналу са брисањем (енгл. *Binary Erasure Channel, BEC*), перформансе контроле грешака могу бити веома добре уз изузетно ниску комплексност хардверских реализација. Стога су честе примене описаних алгоритама у меморијама. Нешто више о перформансама декодера заснованим на тврдим одлукама приказано је у одељку 2.4.3.

2.4.2. ДЕКОДОВАЊЕ ЗАСНОВАНО НА РАЗМЕНИ МЕКИХ ПОРУКА

Са циљем побољшања перформанси контроле грешака, уместо размене бинарних порука, декодовање се може реализовати разменом вишебитних целобројних порука или порука представљених реалним бројевима. Ове такозване меке поруке представљају одређене вероватноће, па одатле и оригинални назив алгоритма: алгоритам пропагације вероватноћа или алгоритам пропагације веродостојности (енгл. *belief-propagation*). У основи, и њега је предложио Галагер [21], а касније су Меккеј и Нил и формално повезали оригинални Галагеров алгоритам са алгоритмом пропагације веродостојности [22]. Овај алгоритам, примењен на итеративно декодовање LDPC кодова назива се још и алгоритмом сумирања и производа (енгл. *Sum-Product Algorithm, SPA*).

Као и код сваког алгоритма размене порука, потребно је иницијализовати варијабилне чворове. Њихова почетна стања, уместо раније поменутих тврдих одлука, представљају априорне вероватноће да су одговарајући послати бити нуле или јединице. У пракси се много чешће користи верзија овог алгоритма која ради у логаритамском домену којим се постиже смањење рачунске комплексности. Због тога се сваки варијабилни чвор v иницијализује логаритмом односа условних вероватноћа да је одговарајући послати бит једнак нули и да је послати бит једнак јединици ако је примљен сигнал y_v. Вредности логаритма односа вероватноћа (енгл. *Logarithm-Likelihood Ratio, LLR*) представљају стања варијабилних чворова и ажурирају се при свакој итерацији декодовања. У случају да се декодује секвенца из канала у коме делује бели Гаусов шум варијансе σ^2 и да су бити мапирани у BPSK (*Binary Phase-Shift Keying*) симболе, те вредности су

$$LLR_v^{(0)} = \ln \frac{Q_v^0}{Q_v^1} = \ln \frac{P\left(b_v = 0 \mid y_v\right)}{P\left(b_v = 1 \mid y_v\right)} = \ln \frac{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_v - 1)^2}{2\sigma^2}}}{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_v + 1)^2}{2\sigma^2}}} = \frac{2y_v}{\sigma^2}, \tag{41}$$

где је са $Q_v^b = P\left(b_v = b \mid y_v\right)$ обележена условна вероватноћа да је вредност бита варијабилног чвора b_v једнака b, под условом да је примљена вредност из канала y_v. Као и раније, иницијализоване вредности уједно представљају и прве поруке које варијабилни чворови шаљу контролним, тј.

$$M_{v\to c}^{(0)} = LLR_v^{(0)} = \ln \frac{Q_{v\to c}^{0(0)}}{Q_{v\to c}^{1(0)}}, \forall v \in \{1, 2, \ldots, n\}, \forall c \in \{1, 2, \ldots, m\}. \tag{42}$$

На овај начин се избегава слање две вероватноће, јер поруку представља њихов однос чиме се уклања потреба за чувањем великог броја додатних порука.

Контролни чворови треба да одговоре порукама које представљају вероватноће да су задовољене једначине провере парности које се у њима израчунавају. Како би се смањио број порука које путују по графу, порука која се шаље једном варијабилном чвору треба да буде однос вероватноће да је задовољена одговарајућа провера парности, ако тај варијабилни чвор има вредност нула, и вероватноће да је задовољена провера парности, ако тај варијабилни чвор има вредност један. Следи извођење израза за овај однос.

Провера парности је задовољена ако је број јединица који улази у контролну суму паран. Галагер је показао да је вероватноћа појављивања парног броја јединица у низу од m бита, под условом да је вероватноћа да k-ти бит има вредност један p_k, следећа:

$$\frac{1}{2} + \frac{1}{2} \prod_{k=1}^{m} (1 - 2p_k). \tag{43}$$

Контролни чворови треба да кажу коју вредност треба да има варијабилни чвор да би провера парности била задовољена. Стога је вероватноћа са којом контролни чвор сматра да варијабилни чвор треба да буде једнак нули, а да је провера парности задовољена, дата следећим изразом:

$$R_{c\to v}^0 = \frac{1}{2} + \frac{1}{2} \prod_{v' \in V_c \setminus v} \left(1 - 2Q_{v'\to c}^1\right). \tag{44}$$

Дакле, та вероватноћа представља вероватноћу да је број јединица у свим његовим суседним чворовима паран. С обзиром на то да важи да је $R_{c\to v}^0 + R_{c\to v}^1 = 1$ и $Q_{v\to c}^0 + Q_{v\to c}^1 = 1$, као и да је

$$\tanh\left(\frac{1}{2}\ln\left(\frac{p_0}{p_1}\right)\right) = p_0 - p_1 = 1 - 2p_1, \tag{45}$$

за $p_0 + p_1 = 1$, из израза (44) се добија

$$\tanh\left(\frac{1}{2}M_{c\to v}\right) = \tanh\left(\frac{1}{2}\ln\left(\frac{R_{c\to v}^0}{R_{c\to v}^1}\right)\right) = \prod_{v' \in V_c \setminus v} \tanh\left(\frac{1}{2}\ln\left(\frac{Q_{v'\to c}^0}{Q_{v'\to c}^1}\right)\right) = \prod_{v' \in V_c \setminus v} \tanh\left(\frac{1}{2}M_{v'\to c}\right), \tag{46}$$

па је коначно

$$M_{c\to v} = 2\tanh^{-1}\left(\prod_{v'\in V_c\setminus v}\tanh\left(\frac{1}{2}M_{v'\to c}\right)\right). \tag{47}$$

Додатно, могу се раздвојити знаци и апсолутне вредности порука варијабилних чворова, па се уз логаритмовање производа тако добија израз

$$M_{c\to v} = \left(\prod_{v'\in V_c\setminus v}\operatorname{sgn}\left(M_{v'\to c}\right)\right)\cdot 2\tanh^{-1}\ln^{-1}\left(\sum_{v'\in V_c\setminus v}\ln\left(\tanh\left(\frac{1}{2}\left|M_{v'\to c}\right|\right)\right)\right), \tag{48}$$

чиме је производ из (47) замењен сумом што олакшава имплементацију[2]. На крају, израчунавање порука контролних чворова може се записати у још компактнијој форми [7]

$$M_{c\to v} = \left(\prod_{v'\in V_c\setminus v}\operatorname{sgn}\left(M_{v'\to c}\right)\right)\cdot \Phi^{-1}\left(\sum_{v'\in V_c\setminus v}\Phi\left(\left|M_{v'\to c}\right|\right)\right), \tag{49}$$

где је $\Phi(x) = -\ln\left(\tanh\left(x/2\right)\right)$ и важи да је $\Phi^{-1}(x) = \Phi(x)$.

Ажурирање варијабилних чворова подразумева два корака: ажурирање стања и генерисање нових порука ка контролним чворовима. Нова вероватноћа да је вредност варијабилног чвора v једнака b, $b\in\{0,\ 1\}$, одређена је са

$$Q_v^{b(it)} = \alpha_v^{(it)} Q_v^{b(it-1)}\prod_{c'\in C_v} R_{c'\to v}^{b(it)} \tag{50}$$

при чему је са it обележен редни број текуће итерације, а параметар $\alpha_v^{(it)}$ је константа која обезбеђује да је $Q_v^{0(it)} + Q_v^{1(it)} = 1$. У логаритамском домену је

$$LLR_v^{(it)} = \ln\frac{Q_v^{0(it)}}{Q_v^{1(it)}} = \ln\frac{\alpha_v^{(it)} Q_v^{0(it-1)}\prod_{c'\in C_v} R_{c'\to v}^{0(it)}}{\alpha_v^{(it)} Q_v^{1(it-1)}\prod_{c'\in C_v} R_{c'\to v}^{1(it)}} = LLR_v^{(it-1)} + \sum_{c'\in C_v} M_{c'\to v}^{(it)}. \tag{51}$$

Порука коју варијабилни чвор *v* шаље контролном чвору *c* је

$$M_{v\to c}^{(it)} = LLR_v^{(it-1)} + \sum_{c'\in C_v\setminus c} M_{c'\to v}^{(it)}. \tag{52}$$

Из (51) и (52) лако је извести и често коришћени израз

$$M_{v\to c}^{(it)} = LLR_v^{(it)} - M_{c\to v}^{(it)}. \tag{53}$$

Након сваке итерације може се проверити тренутна вредност синдрома, па ако је он једнак нула-вектору декодовање се може зауставити. Вредности процена варијабилних чворова одређују се на основу ажурираног стања на крају итерације као

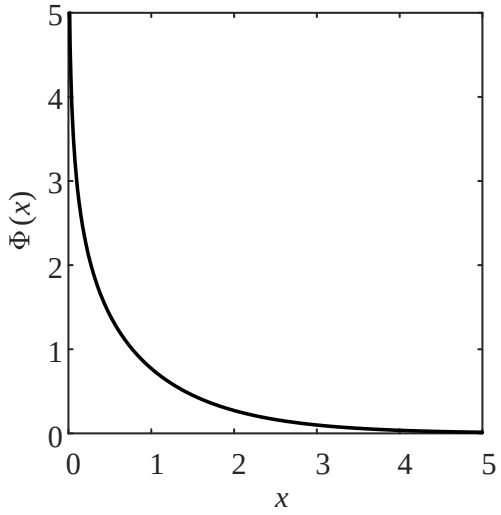
$$b_v^{(it)} = \begin{cases} 1, & \text{ако је } LLR_v^{(it)} < 0 \\ 0, & \text{ако је } LLR_v^{(it)} \geq 0 \end{cases}. \tag{54}$$

Описани алгоритам је у асимптотском случају (за бесконачну дужину кодне речи и велики број итерација) оптималан алгоритам за декодовање у AWGN каналу ако у графу не

[2] Треба водити рачуна да је формално математички $\operatorname{sgn}(0) = 0$, али овде важи једна од следеће две једнакости $\operatorname{sgn}(0) = -1$ или $\operatorname{sgn}(0) = 1$, чешће $\operatorname{sgn}(0) = 1$.

постоје циклуси. С обзиром на то да су кодне речи коначне дужине и да у Танеровом графу LDPC кода постоје циклуси, SPA је субоптималан алгоритам. Због тога се, при пројектовању кода, тежи већим дужинама циклуса. Како дугачке кодне речи по правилу дају боље перформансе контроле грешака од кратких, то је кôд који је први достигао Шенонов капацитет био дужине 10^7 бита, при чему је било потребно више од 1000 итерација декодовања.

Мана SPA алгоритма је комплексност имплементације израчунавања у контролним чворовима. Наиме, хардверска реализација функције $\Phi(x)$ (приказане на слици 2.14) у сваком контролном чвору би захтевала велику количину ресурса. Као заобилазно решење, могуће је апроксимирати је помоћу лукап табела (енгл. *lookup table*). Међутим, функција $\Phi(x)$ даје изузетно велике вредности за мале улазе што захтева коришћење представе бројева са покретним зарезом или изузетно великог броја бита у представи бројева са фиксним зарезом. Једно решење овог проблема је коришћење неуниформне квантизације, код које се поруке које имају мале вредности представљају високом тачношћу, а поруке које имају велике вредности ниском тачношћу [112].



Слика 2.14. График функције Φ(x) коришћене у израчунавањима порука контролних чворова

Широко распрострањене апроксимације SPA алгоритма, које имају значајно мању сложеност израчунавања, засноване су на такозваном *Min-Sum* (скраћено MS) алгоритму [41]. Израчунавања порука контролних чворова код овог алгоритма врше се коришћењем следећег израза:

$$M_{c\to v}=\left(\prod_{v'\in V_c\setminus v}\operatorname{sgn}\left(M_{v'\to c}\right)\right)\cdot\min_{v'\in V_c\setminus v}\left(\left|M_{v'\to c}\right|\right). \tag{55}$$

Идеја је заснована на томе да највећи допринос суми из (49) даје порука најмања по апсолутној вредности. Стога се цела сума мења вредношћу функције $\Phi(x)$ која за аргумент има минимум апсолутних вредности порука варијабилних чворова што поједностављује израз (55). Ипак, тако одређене процене порука контролних чворова имају апсолутне вредности које су веће од оригиналних, тј. њихова поузданост је прецењена. Отуда су перформансе контроле грешака MS декодера значајно лошије од перформанси SPA декодера.

Губитак у перформансама може се делимично надокнадити смањењем апсолутне вредности порука множењем нормализационим параметром α или одузимањем офсета β као у следећим изразима:

$$M_{c\to v}=\left(\prod_{v'\in V_c\setminus v}\operatorname{sgn}\left(M_{v'\to c}\right)\right)\cdot\alpha\min_{v'\in V_c\setminus v}\left(\left|M_{v'\to c}\right|\right), \tag{56}$$

$$M_{c\to v}=\left(\prod_{v'\in V_c\setminus v}\operatorname{sgn}\left(M_{v'\to c}\right)\right)\cdot\max\left(\min_{v'\in V_c\setminus v}\left(\left|M_{v'\to c}\right|\right)-\beta,0\right). \tag{57}$$

Тако добијени алгоритми називају се нормализованим Min-Sum алгоритмом (енгл. *normalized Min-Sum, NMS*) и Min-Sum алгоритмом са офсетом (енгл. *offset Min-Sum, OMS*) респективно [42]. Параметри α и β могу бити константни током целог процеса декодовања, али се могу и динамички мењати у зависности од редног броја текуће итерације [113], од тежине контролних чворова [114] или чак од самих вредности примљених порука варијабилних чворова [115]. Могуће је и модификовати поруке нормализационим и параметром офсета здружено [116]. У литератури се могу наћи и друге модификације Min-Sum алгоритма које укључују више минимума у израчунавање порука (λ-Min алгоритам) [117] или код кога се поруке варијабилних чворова ажурирају само ако не мењају знак у суседним итерацијама, јер се у супротном сматрају за непоуздане (*self-corrected Min-Sum* алгоритам) [118].

2.4.3. ПЕРФОРМАНСЕ АЛГОРИТАМА ДЕКОДОВАЊА

Уобичајени методи за приказ перформанси контроле грешака система са заштитним кодовањем зависе од канала кроз који се преносе информације. На пример, ако се посматрају перформансе система у бинарном симетричном каналу, уобичајено је да се на апсциси приказује вероватноћа прелаза у погрешно стање, а на ординати вероватноћа грешке по информационом биту након декодовања (енгл. *Bit Error Rate, BER*). За AWGN канал прикази се разликују од случаја до случаја, али се уместо вероватноће прелаза на апсциси приказује неки облик односа сигнал-шум. На ординати се може приказати и вероватноћа грешке по кодној речи, тј. вероватноћа да је погрешан макар један бит у декодованим информационим битима који одговарају једној кодној речи. Вероватноћа грешке по кодној речи се у различитим применама различито назива, па се рецимо у DVB стандардима кодна реч поистовећује са фрејмом (енгл. *frame*) одакле се добија израз „вероватноћа грешке по фрејму" (енгл. *Frame Error Rate, FER*), док се у 5G NR стандарду користи израз „вероватноћа грешке по кодном блоку" (енгл. *BLock Error Rate, BLER*).

На слици 2.15 приказане су перформансе контроле грешака неколико раније поменутих алгоритама декодовања за Танеров квазициклични код $(155, 64)$ [73] у BSC каналу. Очигледан је утицај примене различитих алгоритама декодовања на вероватноћу грешке. SPA, као најкомплекснији алгоритам, очекивано даје најмању вероватноћу грешке, док алгоритми који су засновани на тврдим одлукама дају највећу вероватноћу грешке.

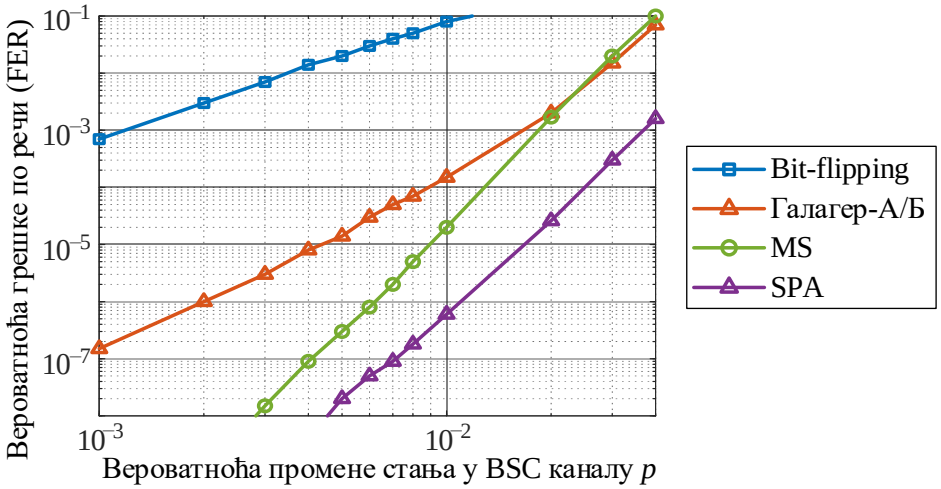
У овој дисертацији, перформансе контроле грешака пројектованог декодера тестиране су у AWGN каналу у коме се вероватноћа грешке приказује у зависности од вредности односа сигнал-шум. Један од начина за приказ односа сигнал-шум је коришћењем односа снаге корисног сигнала у каналу и снаге шума ($P_\mathrm{s}/P_\mathrm{n}$). Раније уведена скраћеница SNR обично има ово значење. Међутим, однос снага корисног сигнала и шума често није погодан за објективну анализу перформанси кода и декодера. Стога се користе и друге представе као што су енергија по симболу подељена са спектралном густином снаге шума (E_s/N_0 , где је $N_0=kT$ за термални шум, а k је Болцманова константа), затим енергија по биту подељена са спектралном густином снаге шума (E_b/N_0) и енергија по информационом биту подељена са

спектралном густином снаге шума. Веза између односа снага сигнала и шума и односа E_s/N_0 може се изразити са

$$\frac{E_s}{N_0} = \frac{P_s}{P_n}\frac{B}{\Phi_s}, \tag{58}$$

где је B пропусни опсег канала, а Φ_s симболски проток [7]. Однос енергије по биту и енергије по симболу зависи од типа модулације, тј. од броја бита у једном симболу M, па на основу тога важи

$$\frac{E_b}{N_0} = \frac{1}{M}\frac{E_s}{N_0}. \tag{59}$$



Слика 2.15. Перформансе контроле грешака различитих алгоритама декодовања у BSC каналу за Танеров код (155, 64) (подаци преузети из [65])

Вероватноћа грешке и кодованог и некодованог система директно зависи од односа E_b/N_0. Поред тога, из (58) је јасно да се енергија по биту може повећати смањењем брзине преноса уз задржавање ширине пропусног опсега. Дакле, смањењем протока информација, може се смањити вероватноћа грешке при преносу [7].

За поређење система са заштитним кодовањем и система без заштитног кодовања, потребно је увести још неколико појмова. С обзиром на то да се информациони проток кодованог система смањује са смањењем кодног количника, не би било адекватно поређење перформанси кодованог и некодованог система ако се оба система посматрају за однос E_b/N_0, добијен на излазу из канала. Због тога је уведен појам енергије по информационом биту, која је од енергије по кодном биту $1/R$ пута мања. На основу тога може се написати и веза два односа E_b/N_0, где је E_b енергија по информационом, односно, кодном биту:

$$\left(\frac{E_b}{N_0}\right)_i = R\left(\frac{E_b}{N_0}\right)_c. \tag{60}$$

Дакле, кодовани и некодовани систем, као и различите системе са заштитним кодовањем, треба поредити за исти однос E_b/N_0, где је E_b енергија по информационом биту, јер се тада сви системи посматрају за случај када имају једнаке информационе протоке [7]. Због тога је у наставку, осим ако другачије није наглашено, са E_b обележавана искључиво енергија по информационом биту.

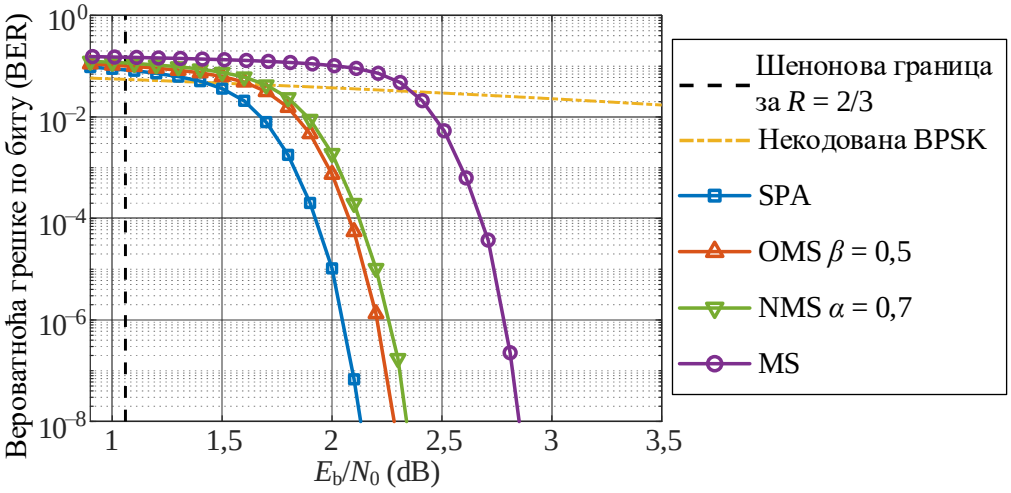
Са циљем јаснијег приказа перформанси контроле грешака, за кодоване системе се дефинише величина под називом кодни добитак (енгл. *coding gain*) [7]:

$$G\left[\mathrm{dB}\right]=10\log\frac{\left(E_\mathrm{b}/N_0\right)_\mathrm{nekodovano}}{\left(E_\mathrm{b}/N_0\right)_\mathrm{i,kodovano}},\tag{61}$$

где је са $\left(E_\mathrm{b}/N_0\right)_\mathrm{nekodovano}$ обележен однос E_b/N_0 некодованог система, а са $\left(E_\mathrm{b}/N_0\right)_\mathrm{i,kodovano}$ однос E_b/N_0 кодованог система. Кодни добитак и однос E_b/N_0 користе се при анализи перформанси система у једном корисничком сценарију. Поред тога, погодно је користити ове параметре приликом пројектовања кодова или декодера за конкретан сценарио ради одабира оних најбољих. Однос E_s/N_0 неретко се користи ако се желе показати перформансе више кодова који су предвиђени да раде у различитим сценаријима, тј. за различите односе снага сигнала и шума. Стога је погодно извести и везу између E_b/N_0 и E_s/N_0, а с обзиром на то да се обе величине често представљају у децибелима, ова веза је изведена у тој представи:

$$\frac{E_\mathrm{b}}{N_0}\left[\mathrm{dB}\right]=\frac{E_\mathrm{s}}{N_0}\left[\mathrm{dB}\right]-10\log\left(M\right)-10\log\left(R\right).\tag{62}$$

Као пример перформанси контроле грешака у AWGN каналу, на слици 2.15 приказана је вероватноћа грешке по информационом биту у зависности од E_b/N_0 за један код из 5G NR стандарда кодног количника 2/3. Перформансе су приказане за различите алгоритме декодовања засноване на SPA алгоритму који се и најчешће користе у AWGN каналу. Примењена је BPSK модулација. На слици је приказана и вероватноћа грешке по биту за некодовану BPSK модулацију, као и ултимативна Шенонова граница за BPSK и кодни количник 2/3 за коју је теоријски могућ пренос са произвољно малом вероватноћом грешке. Вредност за Шенонову границу преузета је из [119]. Са слике је лако уочити да је SPA супериоран у односу на своје апроксимације. Међутим, *Min-Sum* алгоритам са офсетом и нормализовани *Min-Sum* алгоритам за конкретан код показују губитак мањи од 0,3 dB, али уз значајно мању рачунску комплексност. Уобичајено је да OMS и NMS дају веома сличне перформансе, мада постоје резултати који показују да, у случају када се поруке представљају као бројеви са фиксним зарезом, OMS може дати за нијансу боље резултате у зони мале вероватноће грешке [120], [121].



Слика 2.16. Перформансе контроле грешака алгоритама декодовања заснованих на SPA алгоритму у AWGN каналу за код (12672, 8448) из 5G NR стандарда и 20 итерација декодовања

2.4.4. СЛОЈЕВИТО ДЕКОДОВАЊЕ

У до сада посматраним алгоритмима декодовања заснованих на размени порука, у једној итерацији декодовања сви варијабилни чворови свим контролним чворовима симултано шаљу поруке. Такође, сви контролни чворови истовремено шаљу поруке свим варијабилним чворовима. Стога се овакав начин декодовања назива симултаним декодовањем (енгл. *simultaneous decoding schedule* или *flooding schedule*) [24], [122]. Међутим, могуће је урадити декодовање и ако се једна итерација подели на подитерације, а у свакој подитерацији само одређени број чворова размењује поруке. На тај начин варијабилни чворови чешће ажурирају своја стања, чиме се постиже бржа конвергенција алгоритма, тј. иста вероватноћа грешке за мањи број итерација. Уобичајено је да се контролна матрица подели на делове (слојеве) и да се у свакој подитерацији процесира један њен део. Због тога се овакав начин декодовања назива слојевитим декодовањем (енгл. *layered decoding schedule*) [43].

Квазициклични кодови су веома погодни за слојевито декодовање јер им је контролна матрица инхерентно подељена на јасно одвојиве целине. Уобичајено је да један слој има Z врста или Z колона. Ако се контролна матрица дели по врстама, такво декодовање се назива слојевитим декодовањем по врстама (енгл. *row-based layered decoding*), а ако се матрица дели по колонама, такво декодовање се назива слојевитим декодовањем по колонама (енгл. *column-based layered decoding*) [123]. Слојевито декодовање по врстама је рачунски мање комплексно и ефикасније користи меморијске ресурсе од декодовања по колонама, па се и чешће користи [124], [125].

На слици 2.17 приказан је поступак слојевитог декодовања по врстама на примеру једног квазицикличног кода. Чворови Танеровог графа су подељени у групе. Групе су повезане линијама које представљају скуп грана графа пермутованих тако да одговарају пермутационој матрици квазицикличног кода. Као и код сваког итеративног декодовања, варијабилни чворови се иницијализују LLR вредностима добијеним на основу сигнала примљених из канала. Затим само једна група варијабилних чворова шаље поруке само одређеној групи контролних чворова. Групе су одређене првим слојем контролне матрице (осенчен плавом бојом у примеру). Најчешће, у једној подитерацији сваки варијабилни чвор шаље само једну поруку једном од контролних чворова. Ако код није квазицикличан, то не мора да буде случај. Контролни чворови, на основу примљених порука, генеришу одговоре на исти начин као и код симултаног декодовања, коришћењем функције одређене алгоритмом декодовања. Варијабилни чворови, на основу одговора, ажурирају своја стања (LLR вредности), чиме се завршава прва подитерација. У свакој наредној подитерацији понавља се исти поступак са другом групом контролних чворова.

Поруке варијабилних чворова се израчунавају као у (53), при чему за стања варијабилних чворова треба узети најсвежије LLR вредности, тј. поруке треба израчунати као

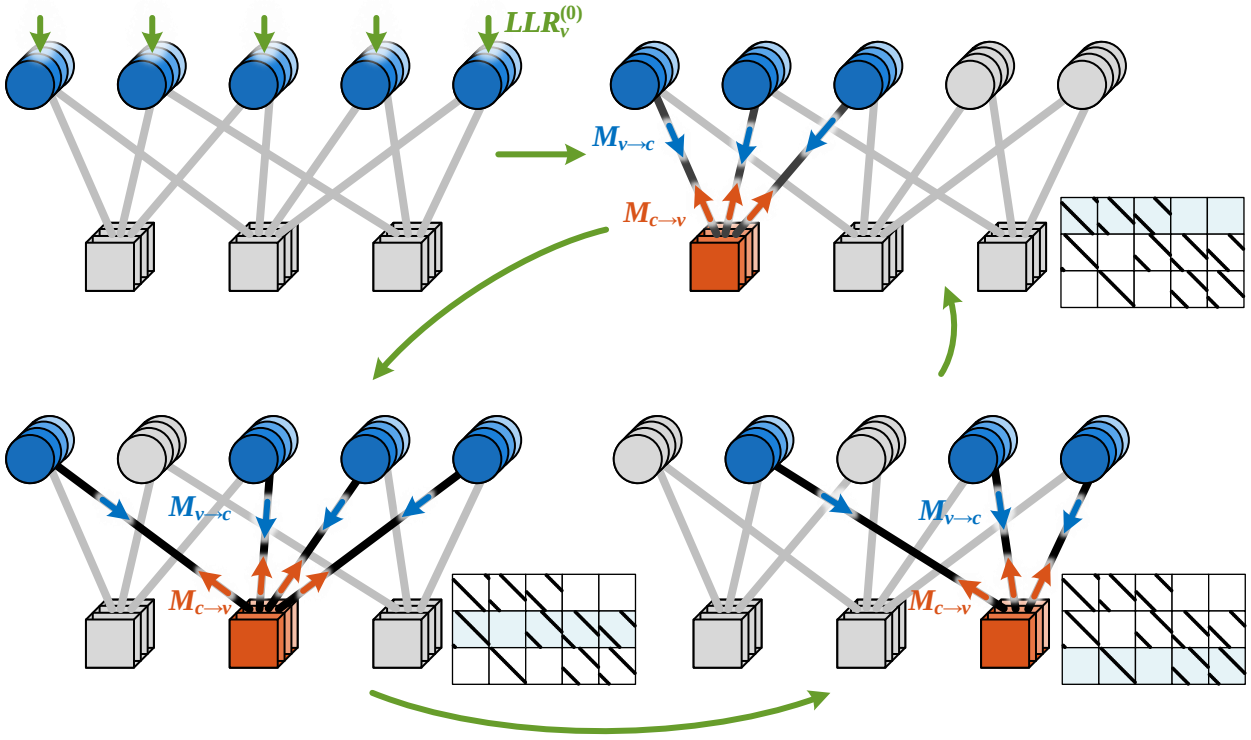
$$M_{v\to c}^{(it)} = LLR_v^{(it-1,l)} - M_{c\to v}^{(it)}, \tag{63}$$

где је са $LLR_v^{(it-1,l)}$ обележена вредност након итерације $it-1$ и након слоја l у текућој итерацији it. Сви доприноси контролних чворова, повезаних на варијабилни чвор v, а који су настали у току једне целе претходне итерације, већ су садржани у вредности $LLR_v^{(it-1,l)}$. То је последица чињенице да се стања варијабилних чворова ажурирају након сваког слоја. Након ажурирања контролних чворова (по једног за сваки варијабилни чвор), потребно је њихове доприносе уврстити у стања варијабилних чворова. Кључна разлика у односу на симултано декодовање настаје управо овде. С обзиром на то да је сваки варијабилни чвор у једном слоју повезан са тачно једним контролним чвором, поруке варијабилних чворова се не ажурирају. Ажурираће се коришћењем израза (63) тек након што се процесирају сви

наредни слојеви, тј. тек у наредној итерацији. Међутим, узимајући у обзир израз (63), поруке варијабилних чворова се могу искористити за ажурирање LLR вредности и то као

$$LLR_v^{(it-1,l+1)} = M_{v\to c}^{(it)} + M_{c\to v}^{(it+1)}. \tag{64}$$

Приметити да се поруке по једној грани графа мењају само једанпут по итерацији, па нема потребе за посебним обележавањем њихових вредности након подитерације l, што наравно није случај са LLR вредностима.



Слика 2.17. Поступак слојевитог декодовања квазицикличних кодова када се слојеви деле по врстама

Ради прегледнијег приказа разлика између симултаног и слојевитог декодовања, на слици 2.18 је дат псеудокод оба метода. Иако су сва израчунавања приказана у псеудокоду секвенцијална, у хардверским реализацијама се многе петље могу израчунавати паралелно или делимично паралелно. Код симултаног декодовања је чак природно да сви контролни чворови рачунају поруке у паралели, што важи и за варијабилне чворове. Могућност за паралелизацију зависи од структуре кода и од доступних хардверских ресурса. Код слојевитог декодовања пак, има смисла да сви контролни чворови из једног слоја примају поруке које паралелно генеришу варијабилни чворови, а затим на основу њих паралелно рачунају своје поруке. Број паралелних операција зависи од структуре кода, пре свега од броја контролних чворова у једном слоју.

Треба нагласити да се слојевитим декодовањем добија бржа конвергенција алгоритма, али је остварива вероватноћа грешке иста као и код симултаног декодовања када се дозволи произвољно велики број итерација. Ово понашање је приказано на слици 2.19 на примеру кода (12672, 8448) из 5G NR стандарда. Када је максималан број итерација мали, слојевито декодовање даје значајно боље перформансе од симултаног, али се перформансе изједначавају повећањем максималног броја итерација.

Алгоритам 1: Симултано декодовање

Улази: контролна матрица **H**,
иницијалне вредности $\mathbf{LLR}^{(0)}$

Иницијализација:

for $v = 1 : n$
- $LLR_v = LLR_v^{(0)}$
- **for** $c \in C_v$
  - $M_{c\to v} = 0$

$it = 1$

Декодовање:

while $it \le it_{max}$ & $\mathbf{s} \neq \mathbf{0}$
- **for** $c = 1 : m$
  - $M_{c\to v} = f\left(\left\{M_{v'\to c} \mid v' \in V_c \setminus v\right\}\right)$
- **for** $v = 1 : n$
  - $LLR_v = LLR_v + \sum_{c' \in C_v} M_{c'\to v}$
  - $M_{v\to c} = LLR_v - M_{c\to v}$
  - $x_v = \dfrac{1-\mathrm{sgn}\left(LLR_v\right)}{2}$
- Израчунавање синдрома: $\mathbf{s} = \mathbf{x}\cdot\mathbf{H}^{\mathrm{T}}$
- $it = it + 1$

Излаз: вектор декодованих бита **x**

Алгоритам 2: Слојевито декодовање

Улази: контролна матрица **H**,
иницијалне вредности $\mathbf{LLR}^{(0)}$

Иницијализација:

for $v = 1 : n$
- $LLR_v = LLR_v^{(0)}$
- **for** $c \in C_v$
  - $M_{c\to v} = 0$

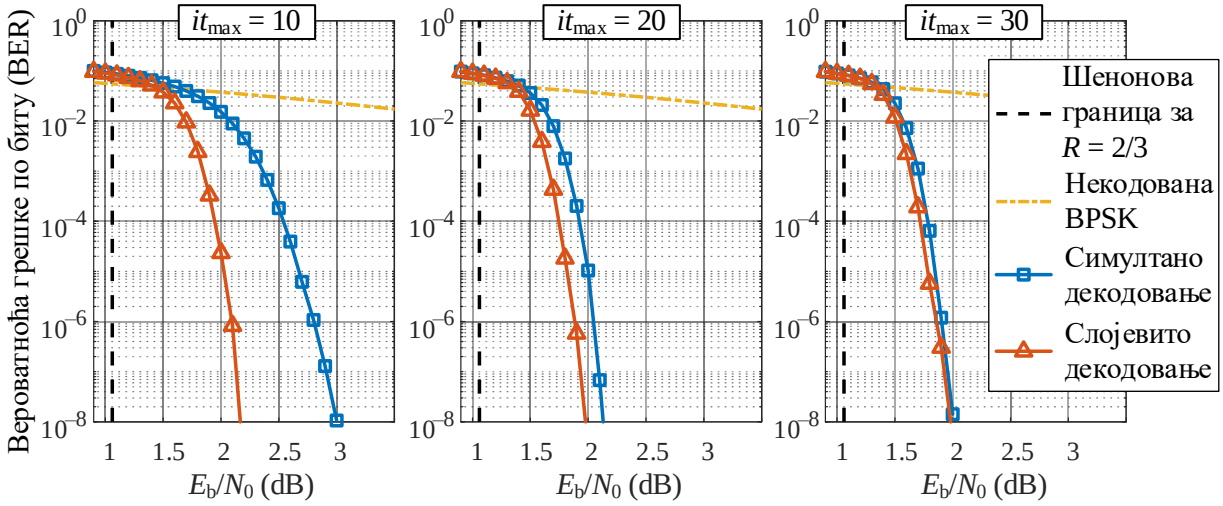
$it = 0$

Декодовање:

while $it < it_{max}$ & $\mathbf{s} \neq \mathbf{0}$
- **for** $l = 1 : (m/Z)$
  - **for** $c = l\cdot Z+1 : (l+1)Z$
    - **for** $v \in V_c$
      - $M_{v\to c} = LLR_v - M_{c\to v}$
    - **for** $v \in V_c$
      - $M_{c\to v} = f\left(\left\{M_{v'\to c} \mid v' \in V_c \setminus v\right\}\right)$
      - $LLR_v = M_{v\to c} + M_{c\to v}$
  - **for** $v = 1 : n$
    - $x_v = \dfrac{1-\mathrm{sgn}\left(LLR_v\right)}{2}$
  - $\mathbf{s} = \mathbf{x}\cdot\mathbf{H}^{\mathrm{T}}$
  - $it = it + 1$

Излаз: вектор декодованих бита **x**

Слика 2.18. Псеудокод алгоритама симултаног и слојевитог декодовања



Слика 2.19. Перформансе контроле грешака симултаног и слојевитог декодовања SPA алгоритмом у AWGN каналу за 5G NR код (12672, 8448) за различит максималан број итерација декодовања

3. ФЛЕКСИБИЛНИ КОДЕР LDPC КОДОВА ЗА 5G СТАНДАРД

Први део доприноса дисертације даје решење за ефикасну хардверску реализацију LDPC кодера за 5G NR стандард. У овом поглављу је описан ефикасан алгоритам кодовања 5G NR кодова заснован на супституцији којим се омогућава флексибилна хардверска реализација. Затим су анализирани различити приступи за хардверску реализацију алгоритма. На основу ове анализе, предложена је нова делимично паралелна архитектура кодера којом се постиже велика ефикасност искоришћења хардверских ресурса. Процес кодовања подржан овом архитектуром оптимизован је коришћењем генетичког алгоритма тако да се максимално смањи трајање кодовања. Доприноси приказани у овом поглављу презентовани су у раду [58]. Поред 5G NR стандарда, приказани резултати могу се применити и на друге кодове. Више детаља о тим применама дато је у потпоглављу 3.5.

3.1. АЛГОРИТАМ ЗА ЕФИКАСНО КОДОВАЊЕ 5G КОДОВА

Стандард 5G NR предвиђа коришћење изузетно великог броја кодова који се, у зависности од тренутних услова у каналу и потреба корисника, могу мењати у току рада. Како би се омогућио рад са свим спектралним ефикасностима и дужинама кодне речи, кодер мора да подржи укупно 102 различита кода која су изведена из две матрице основног графа, не рачунајући скраћења контролне матрице. Овако велика флексибилност се не може остварити техникама кодовања које захтевају множења са густим матрицама без изузетно велике хардверске комплексности. Због тога је у овој дисертацији коришћено кодовање засновано на супституцији, с обзиром на то да је структура LDPC кодова из 5G NR стандарда, описана у одељку 2.2.6.3, погодна за примену овог метода.

Кодовање 5G NR кодова засновано на супституцији, а које је вишеструко ефикасније од директног метода и метода Ричардсона и Урбанкеа, приказано је раније у раду [106], за једну групу кодова изведених из основног графа 1. Поступак кодовања предвиђа израчунавање основних контролних бита коришћењем горњег дела контролне матрице решавањем система једначина у $GF(2)$, на начин веома сличан кодовању Wi-Fi кодова описаном у потпоглављу 2.3.5. Након тога, додатни контролни бити се израчунавају помоћу доњег дела контролне матрице простим сабирањем група кружно померених информационих бита и основних контролних бита у $GF(2)$. У овој дисертацији је поменути метод проширен на преостале кодове изведене из основног графа 1 и на кодове изведене из основног графа 2, како би се подржала флексибилност касније хардверске реализације. Алгоритам је детаљније описан у наставку.

Слично кодовању за Wi-Fi стандард, да би се добили бити парности, потребно је решити следећи систем једначина:

$$\mathbf{H}\mathbf{x}^{\mathrm{T}} = \mathbf{0} \Rightarrow \begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{0} \\ \mathbf{C} & \mathbf{D} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{i}^{\mathrm{T}} \\ \mathbf{p}_{\mathrm{c}}^{\mathrm{T}} \\ \mathbf{p}_{\mathrm{a}}^{\mathrm{T}} \end{bmatrix} = \mathbf{0}. \tag{65}$$

Наведени систем се може поделити на два мања система једначина, од којих се из првог могу израчунати основни контролни бити, а из другог додатни контролни бити:

$$\begin{bmatrix} \mathbf{Q}_{\mathrm{A}_{1,1}} & \mathbf{Q}_{\mathrm{A}_{1,2}} & \cdots & \mathbf{Q}_{\mathrm{A}_{1,k_{\mathrm{b}}}} \\ \mathbf{Q}_{\mathrm{A}_{2,1}} & \mathbf{Q}_{\mathrm{A}_{2,2}} & \cdots & \mathbf{Q}_{\mathrm{A}_{2,k_{\mathrm{b}}}} \\ \mathbf{Q}_{\mathrm{A}_{3,1}} & \mathbf{Q}_{\mathrm{A}_{3,2}} & \cdots & \mathbf{Q}_{\mathrm{A}_{3,k_{\mathrm{b}}}} \\ \mathbf{Q}_{\mathrm{A}_{4,1}} & \mathbf{Q}_{\mathrm{A}_{4,2}} & \cdots & \mathbf{Q}_{\mathrm{A}_{4,k_{\mathrm{b}}}} \end{bmatrix} \times \begin{bmatrix} \mathbf{i}_1^{\mathrm{T}} \\ \mathbf{i}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{i}_{k_{\mathrm{b}}}^{\mathrm{T}} \end{bmatrix} + \begin{bmatrix} \mathbf{Q}_{\mathrm{B}_{1,1}} & \mathbf{Q}_{\mathrm{B}_{1,2}} & \mathbf{Q}_{\mathrm{B}_{1,3}} & \mathbf{Q}_{\mathrm{B}_{1,4}} \\ \mathbf{Q}_{\mathrm{B}_{2,1}} & \mathbf{Q}_{\mathrm{B}_{2,2}} & \mathbf{Q}_{\mathrm{B}_{2,3}} & \mathbf{Q}_{\mathrm{B}_{2,4}} \\ \mathbf{Q}_{\mathrm{B}_{3,1}} & \mathbf{Q}_{\mathrm{B}_{3,2}} & \mathbf{Q}_{\mathrm{B}_{3,3}} & \mathbf{Q}_{\mathrm{B}_{3,4}} \\ \mathbf{Q}_{\mathrm{B}_{4,1}} & \mathbf{Q}_{\mathrm{B}_{4,2}} & \mathbf{Q}_{\mathrm{B}_{4,3}} & \mathbf{Q}_{\mathrm{B}_{4,4}} \end{bmatrix} \times \begin{bmatrix} \mathbf{p}_{\mathrm{c},1}^{\mathrm{T}} \\ \mathbf{p}_{\mathrm{c},2}^{\mathrm{T}} \\ \mathbf{p}_{\mathrm{c},3}^{\mathrm{T}} \\ \mathbf{p}_{\mathrm{c},4}^{\mathrm{T}} \end{bmatrix} = \mathbf{0}, \tag{66}$$

$$\mathbf{p}_{\mathrm{a}}^{\mathrm{T}} = \begin{bmatrix} \mathbf{Q}_{\mathrm{C}_{1,1}} & \mathbf{Q}_{\mathrm{C}_{1,2}} & \cdots & \mathbf{Q}_{\mathrm{C}_{1,k_{\mathrm{b}}}} & \mathbf{Q}_{\mathrm{D}_{1,1}} & \mathbf{Q}_{\mathrm{D}_{1,2}} & \mathbf{Q}_{\mathrm{D}_{1,3}} & \mathbf{Q}_{\mathrm{D}_{1,4}} \\ \mathbf{Q}_{\mathrm{C}_{2,1}} & \mathbf{Q}_{\mathrm{C}_{2,2}} & \cdots & \mathbf{Q}_{\mathrm{C}_{2,k_{\mathrm{b}}}} & \mathbf{Q}_{\mathrm{D}_{2,1}} & \mathbf{Q}_{\mathrm{D}_{2,2}} & \mathbf{Q}_{\mathrm{D}_{2,3}} & \mathbf{Q}_{\mathrm{D}_{2,4}} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{Q}_{\mathrm{C}_{m_{\mathrm{b}}-4,1}} & \mathbf{Q}_{\mathrm{C}_{m_{\mathrm{b}}-4,2}} & \cdots & \mathbf{Q}_{\mathrm{C}_{m_{\mathrm{b}}-4,k_{\mathrm{b}}}} & \mathbf{Q}_{\mathrm{D}_{m_{\mathrm{b}}-4,1}} & \mathbf{Q}_{\mathrm{D}_{m_{\mathrm{b}}-4,2}} & \mathbf{Q}_{\mathrm{D}_{m_{\mathrm{b}}-4,3}} & \mathbf{Q}_{\mathrm{D}_{m_{\mathrm{b}}-4,4}} \end{bmatrix} \times \begin{bmatrix} \mathbf{i}^{\mathrm{T}} \\ \mathbf{p}_{\mathrm{c}}^{\mathrm{T}} \end{bmatrix}. \tag{67}$$

Зависно од структуре подматрице $\mathbf{B}$, за кодове изведене из основног графа 1, систем (66) има једну од следеће две форме:

$$\begin{array}{l|l} \sum_{l=1}^{k_{\mathrm{b}}} \mathbf{Q}_{\mathrm{A}_{1,l}} \mathbf{i}_l^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},1}^{\mathrm{T}(1)} + \mathbf{p}_{\mathrm{c},2}^{\mathrm{T}} = \mathbf{0} & \sum_{l=1}^{k_{\mathrm{b}}} \mathbf{Q}_{\mathrm{A}_{1,l}} \mathbf{i}_l^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},1}^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},2}^{\mathrm{T}} = \mathbf{0} \\ \sum_{l=1}^{k_{\mathrm{b}}} \mathbf{Q}_{\mathrm{A}_{2,l}} \mathbf{i}_l^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},1}^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},2}^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},3}^{\mathrm{T}} = \mathbf{0} & \sum_{l=1}^{k_{\mathrm{b}}} \mathbf{Q}_{\mathrm{A}_{2,l}} \mathbf{i}_l^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},1}^{\mathrm{T}(105 \bmod Z)} + \mathbf{p}_{\mathrm{c},2}^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},3}^{\mathrm{T}} = \mathbf{0} \\ \sum_{l=1}^{k_{\mathrm{b}}} \mathbf{Q}_{\mathrm{A}_{3,l}} \mathbf{i}_l^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},3}^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},4}^{\mathrm{T}} = \mathbf{0} & \sum_{l=1}^{k_{\mathrm{b}}} \mathbf{Q}_{\mathrm{A}_{3,l}} \mathbf{i}_l^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},3}^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},4}^{\mathrm{T}} = \mathbf{0} \\ \sum_{l=1}^{k_{\mathrm{b}}} \mathbf{Q}_{\mathrm{A}_{4,l}} \mathbf{i}_l^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},1}^{\mathrm{T}(1)} + \mathbf{p}_{\mathrm{c},4}^{\mathrm{T}} = \mathbf{0} & \sum_{l=1}^{k_{\mathrm{b}}} \mathbf{Q}_{\mathrm{A}_{4,l}} \mathbf{i}_l^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},1}^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},4}^{\mathrm{T}} = \mathbf{0} \end{array}. \tag{68}$$

Слично поступку описаном у одељку 2.3.5, међурезултати потребни за израчунавање основних контролних бита могу се представити коришћењем израза

$$\boldsymbol{\lambda}_j^{\mathrm{T}} = \sum_{l=1}^{k_{\mathrm{b}}} \mathbf{Q}_{\mathrm{A}_{j,l}} \mathbf{i}_l^{\mathrm{T}}, \; j \in \{1,2,3,4\}. \tag{69}$$

Сабирањем свих једначина из (68), добија се прва група основних контролних бита [58] или прва група основних контролних бита кружно померена за $105 \bmod Z$ позиција, коју је у хардверској реализацији потребно ротирати уназад како би се обезбедио исправан редослед. Даље се из једначина (68) могу израчунати и преостали основни контролни бити [106], и то

$$\begin{array}{l} \mathbf{p}_{\mathrm{c},1} = \sum_{j=1}^{4} \boldsymbol{\lambda}_j; \quad \mathbf{p}_{\mathrm{c},2} = \boldsymbol{\lambda}_1 + \mathbf{p}_{\mathrm{c},1}^{(1)}; \quad \mathbf{p}_{\mathrm{c},4} = \boldsymbol{\lambda}_4 + \mathbf{p}_{\mathrm{c},1}^{(1)}; \quad \mathbf{p}_{\mathrm{c},3} = \boldsymbol{\lambda}_3 + \mathbf{p}_{\mathrm{c},4}; \text{ за } \mathbf{B}_{\mathrm{BG1},1} \\ \hline \mathbf{p}_{\mathrm{c},1}^{(105 \bmod Z)} = \sum_{j=1}^{4} \boldsymbol{\lambda}_j; \quad \mathbf{p}_{\mathrm{c},2} = \boldsymbol{\lambda}_1 + \mathbf{p}_{\mathrm{c},1}; \quad \mathbf{p}_{\mathrm{c},4} = \boldsymbol{\lambda}_4 + \mathbf{p}_{\mathrm{c},1}; \quad \mathbf{p}_{\mathrm{c},3} = \boldsymbol{\lambda}_3 + \mathbf{p}_{\mathrm{c},4}; \text{ за } \mathbf{B}_{\mathrm{BG1},2} \end{array}. \tag{70}$$

За кодове изведене из основног графа 2, систем (66) се може записати на један од следећа два начина:

$$\left.\begin{aligned}
&\sum_{l=1}^{k_\mathrm{b}} \mathbf{Q}_{\mathrm{A}_{1,l}} \mathbf{i}_l^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},1}^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},2}^{\mathrm{T}} = \mathbf{0} \\
&\sum_{l=1}^{k_\mathrm{b}} \mathbf{Q}_{\mathrm{A}_{2,l}} \mathbf{i}_l^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},2}^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},3}^{\mathrm{T}} = \mathbf{0} \\
&\sum_{l=1}^{k_\mathrm{b}} \mathbf{Q}_{\mathrm{A}_{3,l}} \mathbf{i}_l^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},1}^{\mathrm{T}(1)} + \mathbf{p}_{\mathrm{c},3}^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},4}^{\mathrm{T}} = \mathbf{0} \\
&\sum_{l=1}^{k_\mathrm{b}} \mathbf{Q}_{\mathrm{A}_{4,l}} \mathbf{i}_l^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},1}^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},4}^{\mathrm{T}} = \mathbf{0}
\end{aligned}\right| \left|\begin{aligned}
&\sum_{l=1}^{k_\mathrm{b}} \mathbf{Q}_{\mathrm{A}_{1,l}} \mathbf{i}_l^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},1}^{\mathrm{T}(1)} + \mathbf{p}_{\mathrm{c},2}^{\mathrm{T}} = \mathbf{0} \\
&\sum_{l=1}^{k_\mathrm{b}} \mathbf{Q}_{\mathrm{A}_{2,l}} \mathbf{i}_l^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},2}^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},3}^{\mathrm{T}} = \mathbf{0} \\
&\sum_{l=1}^{k_\mathrm{b}} \mathbf{Q}_{\mathrm{A}_{3,l}} \mathbf{i}_l^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},1}^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},3}^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},4}^{\mathrm{T}} = \mathbf{0} \\
&\sum_{l=1}^{k_\mathrm{b}} \mathbf{Q}_{\mathrm{A}_{4,l}} \mathbf{i}_l^{\mathrm{T}} + \mathbf{p}_{\mathrm{c},1}^{\mathrm{T}(1)} + \mathbf{p}_{\mathrm{c},4}^{\mathrm{T}} = \mathbf{0}
\end{aligned}\right. . \tag{71}$$

Слично као и за кодове изведене из основног графа 1, основни контролни бити се добијају из следећих израза:

$$\begin{aligned}
&\mathbf{p}_{\mathrm{c},1}^{(1)} = \sum_{j=1}^{4} \boldsymbol{\lambda}_j; \quad \mathbf{p}_{\mathrm{c},2} = \boldsymbol{\lambda}_1 + \mathbf{p}_{\mathrm{c},1}; \quad \mathbf{p}_{\mathrm{c},3} = \boldsymbol{\lambda}_2 + \mathbf{p}_{\mathrm{c},2}; \quad \mathbf{p}_{\mathrm{c},4} = \boldsymbol{\lambda}_4 + \mathbf{p}_{\mathrm{c},1}; \text{ за } \mathbf{B}_{\mathrm{BG2},1} \\
&\mathbf{p}_{\mathrm{c},1} = \sum_{j=1}^{4} \boldsymbol{\lambda}_j; \quad \mathbf{p}_{\mathrm{c},2} = \boldsymbol{\lambda}_1 + \mathbf{p}_{\mathrm{c},1}^{(1)}; \quad \mathbf{p}_{\mathrm{c},3} = \boldsymbol{\lambda}_2 + \mathbf{p}_{\mathrm{c},2}; \quad \mathbf{p}_{\mathrm{c},4} = \boldsymbol{\lambda}_4 + \mathbf{p}_{\mathrm{c},1}^{(1)}; \text{ за } \mathbf{B}_{\mathrm{BG2},2}
\end{aligned} \tag{72}$$

Додатни контролни бити се за све кодове директно израчунавају из (67) коришћењем следећег израза:

$$\mathbf{p}_{\mathrm{a},j}^{\mathrm{T}} = \sum_{l=1}^{k_\mathrm{b}} \mathbf{Q}_{\mathrm{C}_{j,l}} \mathbf{i}_l^{\mathrm{T}} + \sum_{l=1}^{4} \mathbf{Q}_{\mathrm{D}_{j,l}} \mathbf{p}_{\mathrm{c},l}^{\mathrm{T}}, \; j \in \{1, 2, \ldots, m_\mathrm{b} - 4\}. \tag{73}$$

Сва множења вектора циркулантима представљају кружни померај тих вектора или нула-векторе, па је уместо целог циркуланта довољно чувати само вредности кружног помераја. Ово у многоме олакшава хардверску реализацију и омогућава флексибилност, што је и један од разлога коришћења описаног метода у овој дисертацији.

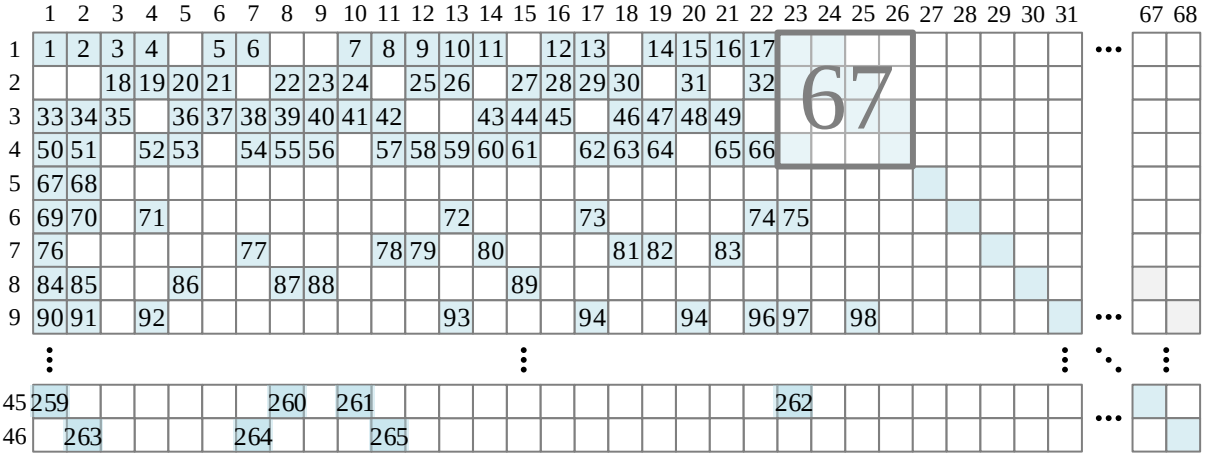
3.2. ОПТИМАЛНИ ПРОЦЕС КОДОВАЊА У ДЕЛИМИЧНО ПАРАЛЕЛНОЈ АРХИТЕКТУРИ КОДЕРА

Алгоритам описан у потпоглављу 3.1 може се имплементирати потпуно паралелно, серијски или делимично паралелно. Као што је приказано у уводу, серијска реализација би дала мале протоке и велико кашњење. Насупрот томе, потпуно паралелна реализација кодера, која треба да подржи све кодове задате стандардом, заузела би изразито велику количину хардверских ресурса, што би додатно узроковало и релативно дугачку критичну путању. Делимично паралелне архитектуре омогућавају компромис између остваренog протока и кашњења, са једне стране, и употребљених хардверских ресурса, са друге. У овом потпоглављу дат је преглед неколико делимично паралелних метода процесирања и њихова анализа са становишта ефикасности искоришћења хардверских ресурса. На основу приказане анализе, предложена је делимично паралелна реализација специфична за 5G NR стандард којом се може повећати проток за кратке кодне речи, а самим тим и ефикасност искоришћења хардверских ресурса. За предложени делимично паралелни метод, редослед процесирања елемената контролне матрице је оптимизован за минимално трајање кодовања, чиме је додатно повећан проток и смањено кашњење кодера.

3.2.1. ДЕЛИМИЧНО ПАРАЛЕЛНЕ АРХИТЕКТУРЕ КОДЕРА

Кључни елементи у архитектури флексибилног кодера су мрежа за кружни померај и „ексили" кола. Мрежа за кружни померај се састоји од једног или више комбинационих елемената, који се називају кружни померачи или ротатори. Узимајући у обзир претходно описани алгоритам декодовања, ротатори кружно померају секвенцу од Z бита. С обзиром на то да величина циркуланта у 5G стандарду варира од 2 до 384, ротатори морају да буду флексибилни и да подрже све дужине улазних вектора предвиђене стандардом. Реализација таквих компоненти није тривијалан задатак, а захтев флексибилности у значајној мери повећава количину хардверских ресурса које они заузимају и то до те мере да су ротатори компоненте које заузимају највише ресурса у читавом систему. Због тога је, у циљу постизања ефикасне хардверске реализације, важно да се кружни померачи максимално користе у току кодовања, тј. да су ретко неактивни.

Делимично паралелно процесирање се може урадити на више различитих начина. Када је на располагању само један ротатор, кодовање се врши серијски, процесирајући један циркулант по циклусу такта. Треба напоменути да је ово и даље делимично паралелан начин кодовања, с обзиром на то да се истовремено (паралелно) израчунава Z контролних сума, иако је овде названо серијским кодовањем. Редослед процесирања циркуланата у таквој конфигурацији приказан је на слици 3.1. Како је контролна матрица кодова изведених из основног графа 2 исте структуре и садржи мањи број циркуланата, редослед је приказан само за кодове изведене из основног графа 1. Бројеви на позицијама циркуланата представљају редне бројеве циклуса такта у коме се одговарајући циркуланти процесирају. За израчунавање међурезултата $\boldsymbol{\lambda}_j$ из (69) потребно је обрадити прве четири врсте пермутационе матрице. На основу израчунатих међурезултата рачунају се основни контролни бити, коришћењем израза (70) или (72). Ово се дешава паралелно, у 67. циклусу такта, при чему се истовремено може процесирати и први циркулант из пете врсте. Израчунавање основних контролних бита могуће је поделити и на више тактова коришћењем проточне обраде. Даље процесирање подразумева ротирање информационих и претходно израчунатих контролних бита, дефинисано циркулантима од пете до последње врсте, и израчунавање „ексили" операције над ротираним векторима.



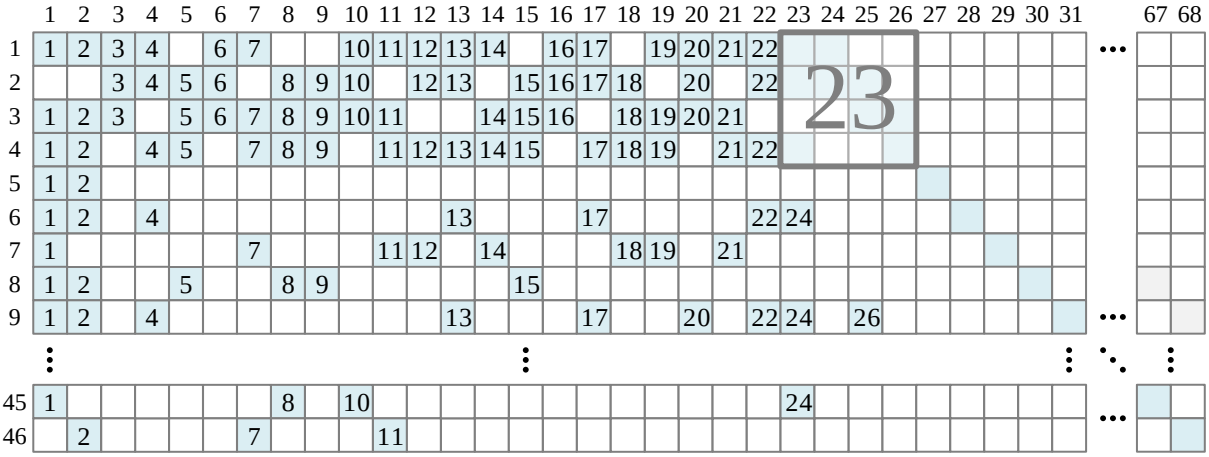
Слика 3.1. Редослед процесирања циркуланата контролне матрице изведене из основног графа 1 при серијском процесирању једног циркуланта по циклусу такта [58]

За серијско кодовање је потребно 265 циклуса такта за кодове изведене из основног графа 1 и 150 циклуса такта за кодове изведене из основног графа 2. Иако је проток таквог приступа најмањи од свих посматраних делимично паралелних приступа у овом одељку,

ефикасност искоришћења хардверских ресурса је највећа, што је последица тога да ротатор искоришћен за кружни померај никада није неактиван.

Сличан принцип је предложен у [104] за кодове из Wi-Fi стандарда. Разлика је у томе што се процесирање обавља по колонама, уместо по врстама, па је потребно чување међурезултата у локалној меморији.

Увођењем већег паралелизма, може се повећати проток и смањити кашњење. На слици 3.2 приказан је метод кодовања у коме се у једном такту паралелно процесира цела колона циркуланата. Метод је предложен у [105] за Wi-Fi кодове, а овде је примењен на кодове из 5G NR стандарда. Након процесирања првих k_b колона, међурезултати $\boldsymbol{\lambda}_j$ су израчунати, па се из њих могу израчунати основни контролни бити, што се у примеру са слике дешава у 23. циклусу такта. Након израчунавања основних контролних бита, додатни контролни бити се добијају процесирањем преостале четири колоне. По колонама паралелно кодовање може се завршити за минимално 27 ($k_{b,BG1}+1+4$) циклуса такта за кодове изведене из основног графа 1, односно за 15 ($k_{b,BG2}+1+4$) циклуса такта за кодове изведене из основног графа 2, што је значајно убрзање у поређењу са серијским кодовањем. Међутим, потребан број ротатора којим се обезбеђује овако велики паралелизам је 46 за кодове основног графа 1 и 42 за кодове основног графа 2. С обзиром на то да кодер треба да подржи све кодове задате стандардом, број ротатора је већи од наведена два.

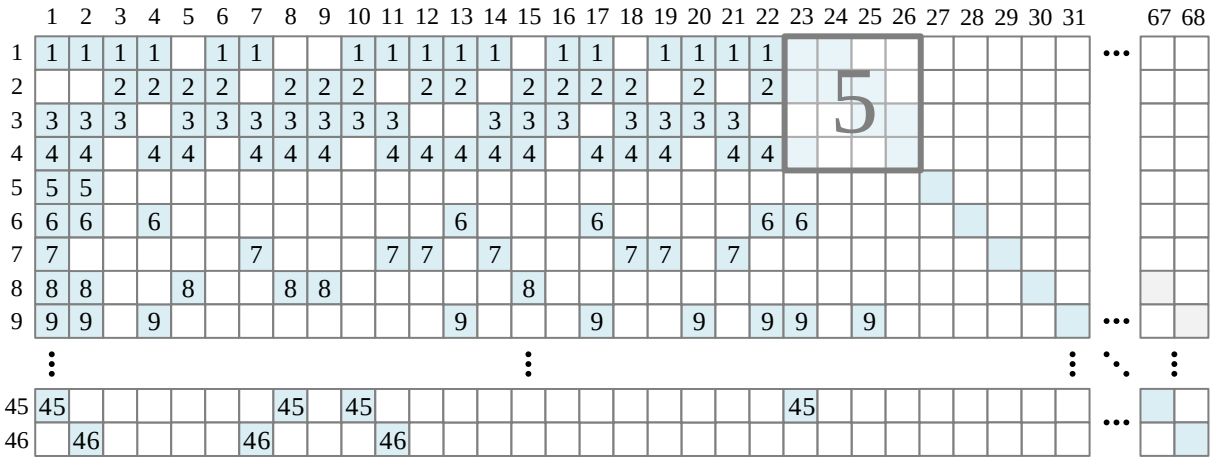


Слика 3.2. Редослед процесирања циркуланата контролне матрице изведене из основног графа 1 при паралелном процесирању свих циркуланта једне колоне у једном циклусу такта [58]

Други метод процесирања са великим паралелизмом је паралелно процесирање по врстама код кога се у једном циклусу такта паралелно процесирају сви циркуланти из једне врсте. Метод је предложен у [103] и [106], а пример је приказан на слици 3.3. За разлику од претходна два метода, овај начин процесирања захтева да су сви информациони бити, а касније и сви основни контролни бити, доступни у регистрима. С обзиром на то да се улазни подаци обично учитавају коришћењем делимично паралелног интерфејса, често је неопходно сачекати да се сви информациони бити са улаза упишу у регистре [106]. Ипак, ако се омогући дупло баферисање коришћењем двоструке банке регистара за улазне бите, ово кашњење се може избећи. Како су сада све кружно померене вредности које улазе у једну суму доступне у исто време, сума се најчешће реализује коришћењем стабла „ексили" кола.

За по врстама паралелно кодовање потребно је минимално 46 ($m_{b,BG1}$) циклуса такта за кодове изведене из основног графа 1, односно 42 ($m_{b,BG2}$) циклуса такта за кодове изведене из основног графа 2, што је значајно већи број циклуса него код по колонама паралелног кодовања. Међутим, потребан број ротатора је мањи, 26 за кодове основног графа 1 и 14 за

кодове основног графа 2. Као и раније, због флексибилности, број потребних ротатора је већи од наведена два.



Слика 3.3. Редослед процесирања циркуланата контролне матрице изведене из основног графа 1 при паралелном процесирању свих циркуланта једне врсте у једном циклусу такта [58]

Један од циљева дисертације је ефикасна хардверска реализација кодера која омогућава највећи проток за најмање заузеће хардверских ресурса, па је стога од интереса упоредити горенаведене технике кодовања. На почетку је потребно проценити проток сваког од метода. Информациони проток кодера се може изразити са

$$T = \frac{k_{\mathrm{b}} Z f_{\mathrm{CLK}}}{N_{\mathrm{CPC}}}, \tag{74}$$

где је f_{CLK} учестаност сигнала такта, а N_{CPC} број циклуса такта потребан за кодовање једне кодне речи (енгл. *Clocks Per Codeword*, *CPC*). Проток се може нормализовати на један циклус такта:

$$T_{\mathrm{norm}} = \frac{T}{f_{\mathrm{CLK}}} = \frac{k_{\mathrm{b}} Z}{N_{\mathrm{CPC}}}, \tag{75}$$

а уобичајена јединица за нормализовани проток је b/cycle (бита по циклусу). Заузеће хардверских ресурса се апроксимативно може изразити као број употребљених ротатора, пошто су то компоненте које заузимају највећу површину на чипу. Стога се апроксимативна ефикасност искоришћења хардверских ресурса може израчунати са

$$E_{\mathrm{approx}} = \frac{T}{N_{\mathrm{rots}}}, \tag{76}$$

где је са N_{rots} обележен број употребљених ротатора. Слично, нормализована ефикасност се добија из

$$E_{\mathrm{norm,approx}} = \frac{T_{\mathrm{norm}}}{N_{\mathrm{rots}}}. \tag{77}$$

Нормализовани параметри описаних архитектура кодера, добијени за кодне речи највеће дужине ($k_{\mathrm{max,BG1}} = k_{\mathrm{b,BG1}} Z_{\mathrm{max}} = 8448$ и $k_{\mathrm{max,BG2}} = k_{\mathrm{b,BG2}} Z_{\mathrm{max}} = 3840$), приказани су у табели 3.1. Као што је и очекивано, већи паралелизам омогућава и веће протоке. Међутим, изразито паралелне архитектуре имају знатно мању ефикасност искоришћења хардверских

ресурса. Ово је последица тога што је велики број ротатора неактиван у току кодовања јер највећи број колона и врста пермутационе матрице није у потпуности попуњен.

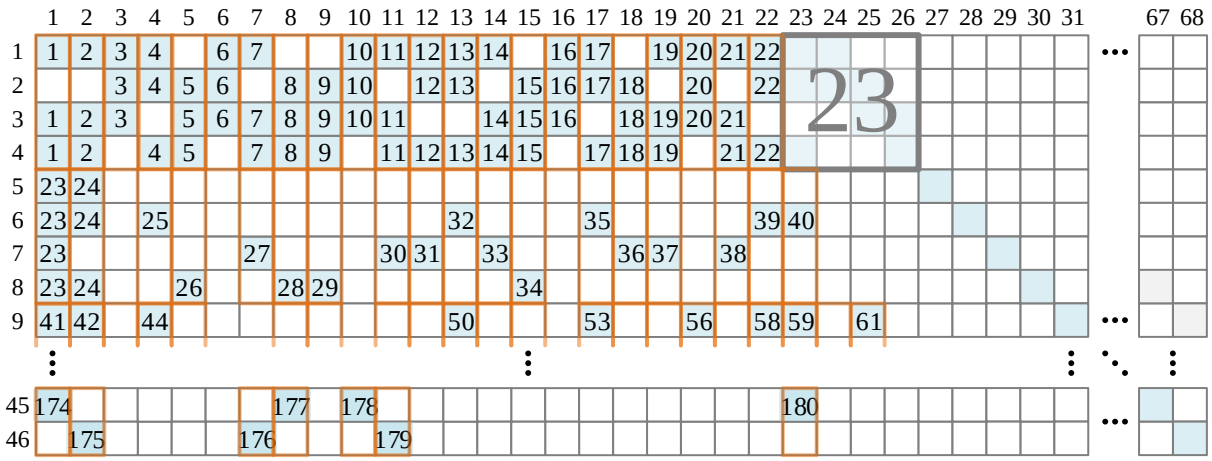
Табела 3.1. Поређење протока и апроксимативне ефикасности искоришћења хардверских ресурса архитектура флексибилних кодера различитих нивоа паралелизма за најдуже кодне речи из 5G NR стандарда [58]

Метод процесирања	N_{CPC}		$T_{\mathrm{norm}}\left(\mathrm{b/cycle}\right)$		N_{rots}	$E_{\mathrm{norm,approx}}\left(\mathrm{b/cycle/rot}\right)$	
	BG1	BG2	BG1	BG2		BG1	BG2
Серијски	265	150	31,9	25,6	1	31,9	25,6
По колонама паралелни	27	15	312,8	257,6	46	6.8	5.6
По врстама паралелни	46	42	184,6	91	26	7.1	3.5

На основу претходне анализе може се закључити да је, за остваривање произвољно великог протока, боље користити више серијских кодера који раде у паралели, него један кодер са већим уграђеним паралелизмом, под условом да је кашњење кодера задовољавајуће. На тај начин се користи много мање хардверских ресурса, што додатно доводи и до енергетски ефикаснијег дизајна. Због тога је приступ у коме се користи само један ротатор усвојен и у дисертацији. Међутим, ротатор је пројектован тако да се може реконфигурисати да ради као два или више независних ротатора у случају када се врши кодовање кодова чија је величина циркуланта мања или једнака 192. Овим је постигнут већи паралелизам при кодовању краћих кодова, него што је то инхерентно предвиђено самим кодом. Логички ресурси ротатора, који иначе не би били активни при кружном померају кратких улазних вектора, користе се за кружни померај других независних вектора. Дизајн таквог ротатора захтева нешто више хардверских ресурса од ротатора који може да ради кружни померај само једног улазног вектора, али се добитак у оствареном протоку може утростручити што је касније и показано. Сам ротатор и архитектура целог кодера приказани су у потпоглављу 3.3, док је овде описан редослед процесирања који је омогућен таквом архитектуром мреже за кружни померај.

У ситуацијама када се кодују кратки кодови, на располагању је већи број ротатора него што је то био случај при класичном серијском кодовању. Стога се серијски метод процесирања може паралелизовати тако да се истовремено обрађује већи број циркуланата. Пример са четири ротатора на располагању, што би одговарало кодовима чија је величина циркуланта $Z \leq 96$, приказан је на слици 3.4. Предложени метод подразумева да се процесирају циркуланти који деле групе информационих бита или основних контролних бита, како би се омогућило само једно читање из меморије за целу групу циркуланата.

Ако су на располагању четири ротатора, предложеном паралелизованом серијском методу је потребно 180 циклуса такта за кодовање кодова изведених из основног графа 1 и 94 циклуса такта за кодове изведене из основног графа 2, што је значајна уштеда у односу на стандардно серијско процесирање. Међутим, са слике 3.4 може се приметити да је честа ситуација да су доступна четири ротатора, али да се само један од њих користи. Прецизније речено, ово понашање је карактеристично за генерисање свих додатних контролних бита. Ово је последица дизајна 5G NR LDPC кодова код којих су суседне врсте у доњем делу пермутационе матрице ортогоналне или квазиортогоналне. Ортогоналне врсте су оне код којих не постоје циркуланти који деле исту групу информационих бита (на пример врсте 45 и 46 са слике 3.4), док су квазиортогоналне врсте оне код којих су једини циркуланти који деле исту групу информационих бита на позицијама 1 и/или 2 (на пример врсте 6 и 7 са слике 3.4). Такав дизајн кодова је одабран са циљем да се избегну хазарди података услед превременог читања који настају при слојевитом декодовању [32].



Слика 3.4. Предложени редослед процесирања циркуланата контролне матрице изведене из основног графа 1 када су доступна четири ротатора [58]

Како би се повећала ефикасност предложеног метода, у дисертацији се процесирање врста не обавља по предефинисаном редоследу. Наиме, шеста и девета врста са слике 3.4 деле велики број информационих бита, тј. имају циркуланте у истим колонама. Значајно већа ефикасност би се остварила ако би се те две врсте процесирале заједно. Стога се предложени метод може оптимизовати са циљем да се што више врста, које имају циркуланте у истим колонама, процесира заједно. У том случају се додатни контролни бити не генеришу по конвенционалном редоследу од прве групе ка последњој, па се редослед мора изменити пре слања излазних података. Међутим, узимајући у обзир то да је уобичајено да се излазни подаци прво чувају у некој врсти меморије, како би се омогућило двоструко баферисање излаза, ово ограничење не утиче додатно на хардверску комплексност. Метод за оптимизацију редоследа процесирања којим се добија минимално време декодовања описан је у одељку 3.2.2.

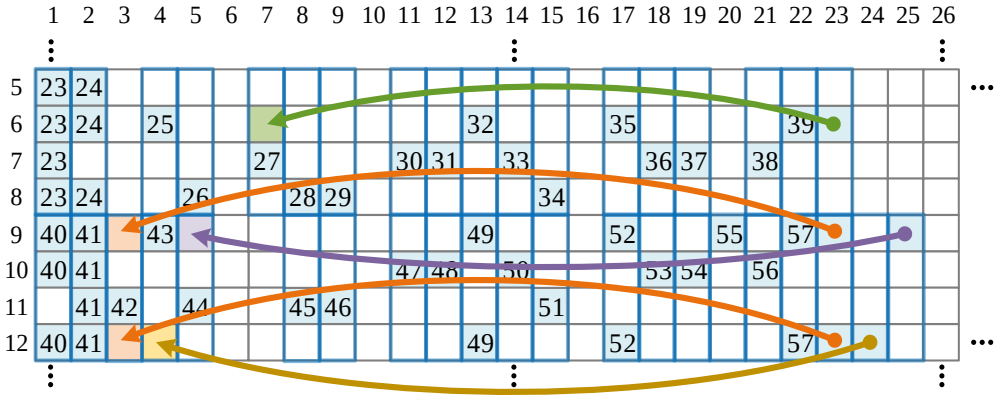
3.2.2. Оптимизација редоследа процесирања

Проналажење оптималног редоследа процесирања врста спада у класу проблема трговачког путника (енгл. *Traveling Salesman Problem, TSP*), који се ефикасно могу решити коришћењем генеричког алгоритма [57], [48]. Генетички алгоритам је раније коришћен за одређивање оптималног редоследа процесирања при слојевитом декодовању LDPC кодова [48]. Оптималан редослед декодовања обезбеђује најмањи број хазарда података услед превременог читања, а примењен је и у овој дисертацији и претходно објављеном раду [49] за исте намене, што је детаљно описано у поглављу 4. Распоред процесирања врста који обезбеђује најмањи број конфликата је најчешће такав да суседне врсте имају веома мали број циркуланата у истим колонама. Међутим, за потребе кодовања оптимизацију треба урадити на потпуно супротан начин. Стога је оптимизациона функција (или фитнес функција), написана тако да фаворизује оне распореде врста који обезбеђују да суседне врсте имају што већи број циркуланата у истим колонама. Резултат оптимизационе функције је број циклуса такта потребан за кодовање целе кодне речи, N_{CPC}, који искључиво зависи од редоследа процесирања врста. Овакав критеријум оптимизације омогућава максимално искоришћење доступних ротатора, што даље обезбеђује најкраће кодовање и највећу ефикасност.

Важно је напоменути да је предвиђено да се процесирање прве четири врсте у пермутационој матрици увек врши на самом почетку кодовања. Ово је неопходно јер су основни контролни бити потребни за касније израчунавање додатних контролних бита. Стога је измена редоследа врста урађена само за врсте почев од пете па до последње.

Поред оптимизације редоследа процесирања врста, урађена је додатна оптимизација обраде циркуланата који одговарају основним битима парности. Наиме, у одељку 3.2.1, приликом израчунавања додатних контролних бита, сви основни контролни бити су кружно померани независно од информационих бита, тј. у одвојеним циклусима такта. Међутим, пошто се, и поред оптимизације, и даље може десити да је неки од ротатора слободан за коришћење, могуће је спојити кружно померање основних контролних бита са кружним померањем информационих бита, али само онда када постоји слободан ротатор. Пример спајања процесирања циркуланата приказан је на слици 3.5. На описани начин губи се потреба за посебним циклусима такта за обраду циркуланата који одговарају основним битима парности. Спајање процесирања бита парности са информационим битима је укључено у оптимизациони поступак модификацијом фитнес функције, тако да се фаворизују распореди процесирања врста код којих је могуће урадити спајање. На овај начин се добија редослед обраде код кога се сви циркуланти који одговарају битима парности процесирају заједно са неким од циркуланата који одговарају информационим битима. Ова измена знатно доприноси убрзању кодовања. Ово је посебно значајно за кодове изведене из основног графа 2. Детаљније анализе приказане су у одељку 3.4.1.

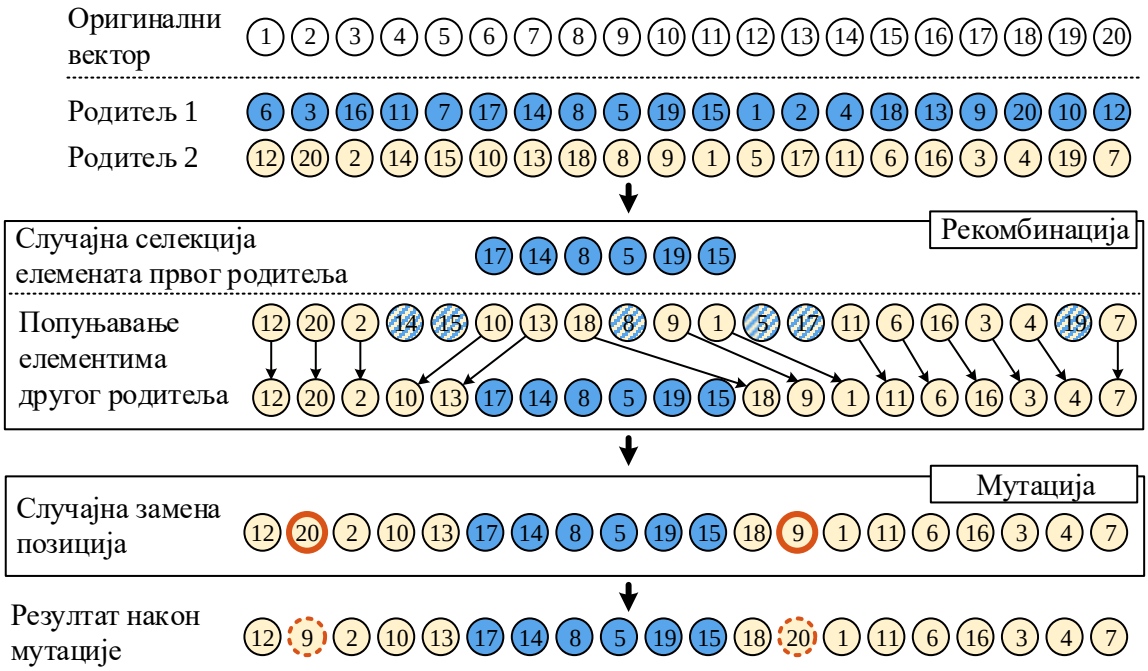
Поступак оптимизације генетичким алгоритмом се састоји од неколико фаза. Прва фаза је иницијализација почетне популације јединки, тј. неке репрезентације њихових гена. У конкретном случају, основна јединка је вектор природних бројева са елементима од 5 до m_b, који представља оригинални распоред процесирања врста. Иницијална популација садржи велики број случајних пермутација основне јединке. Након иницијализације, израчунава се фитнес функција за сваку јединку и на основу добијених резултата селектују се оне јединке из којих се креира наредна генерација популације. Селекција може бити урађена на мноштво различитих начина, на пример случајним одабиром или одабиром јединки које имају најбољу вредност фитнес функције. У дисертацији је извршена селекција на основу најмање вредности фитнес функције. Нова генерација популације се генерише укрштањем или рекомбинацијом (енгл. *crossover*), а након ње и мутацијом случајно одабраних јединки из нове генерације. У процес рекомбинације улазе парови претходно селектованих јединки, а они се могу бирати случајно или детерминистички.



Слика 3.5. Илустрација премештања процесирања циркуланата који одговарају основним контролним битима [58]

На слици 3.6 приказан је пример рекомбинације два случајна вектора чији елементи, због једноставнијег приказа, узимају вредности између 1 и 20. Међутим, пример се лако може проширити за рекомбинацију вектора чији елементи узимају било које вредности. На почетку се из првог вектора (родитеља 1) одабере подсеквенца случајне дужине почев од случајно одабране позиције. Селектована подсеквенца је први део нове секвенце, а поставља се на исте позиције на којима је и била у оригиналном вектору. Остале позиције нове

секвенце попуњавају се елементима другог родитеља, задржавајући оригинални редослед. Они елементи који су једнаки већ постављеним елементима првог родитеља се прескачу, што је неопходно како би се задржала јединственост сваког елемента у вектору. Рекомбинација се понавља све док се не изгенерише цела нова генерација популације. Након тога се, за одређени број случајно изабраних јединки, обавља мутација и то заменом места два случајно одабрана елемента у вектору. Добијена популација се поново уводи у процес селекције, нове рекомбинације и нове мутације како би се добила наредна генерација. Поступак се понавља све док се не дође до предефинисаног, максималног броја генерација. Могуће је зауставити га и раније ако се најбоље решење не поправља након неколико узастопних генерација. Резултати се могу побољшати ако се генетички алгоритам покрене више пута за више различитих случајно генерисаних иницијалних популација.



Слика 3.6. Примери рекомбинације и мутације за векторе од 20 елемената [58]

3.3. Хардверска реализација кодера

У овом потпоглављу је описана флексибилна хардверска имплементација претходно описаног алгоритма кодовања. Алгоритам омогућава реализацију којом се постиже велики ниво паралелизма и за кодове мање дужине ,захваљујући флексибилном подешавању режима рада зависно од величине циркуланта у контролној матрици кода. У предложеној реализацији предвиђено је да се кодови чија је величина циркуланта у опсегу $192 < Z \leq 384$ кодују серијски, тј. процесирањем једног циркуланта по циклусу такта. Међутим, за кодове чија је величина циркуланта у опсегу $96 < Z \leq 192$ предвиђен је већи паралелизам, тј. процесирају се максимално два циркуланта по циклусу такта, док је за преостале кодове предвиђено да се процесирају максимално четири циркуланта по циклусу такта. Циљ је искористити хардверске ресурсе који би иначе били неактивни при серијском кодовању. Подршка за описану флексибилност ипак захтева и одређене додатне хардверске ресурсе, али су процентуална побољшања у протоку вишеструко већа од процента додатних ресурса, па се тиме ефикасност повећава. За $Z \leq 48$ могуће је и додатно повећати број циркуланата који се истовремено процесирају. Међутим, за 5G NR кодове, овај, додатни степен флексибилности захтева повећање заузећа хардверских ресурса, а недовољно повећава

проток, па је то разлог због кога је у овој дисертацији задржано да је максималан број истовремено процесираних циркуланата једнак четири.

С обзиром на то да је кључни блок за успешну реализацију флексибилног кодера мрежа за кружни померај, посебан одељак је посвећен њеном дизајну. Након тога је представљена архитектура самог кодера која користи пројектовану мрежу.

3.3.1. ФЛЕКСИБИЛНА МРЕЖА ЗА КРУЖНИ ПОМЕРАЈ

Архитектура кодера је заснована на флексибилној мрежи за кружни померај која омогућава више режима рада. Како би се подржала раније описана флексибилност, мрежа за кружни померај је пројектована тако да ради као један ротатор ако се кодују дугачки кодови (тј. за $192 < Z \leq 384$), затим као два независна ротатора ако се кодују кодови средње дужине ($96 < Z \leq 192$), или као четири независна ротатора за случај кодовања кратких кодова ($2 \leq Z \leq 96$). Основни градивни блок такве мреже је флексибилни ротатор који подржава све величине циркуланата предвиђене 5G NR стандардом које су мање или једнаке 96. Четири таква блока су искоришћена као независни ротатори или, ако су величине циркуланата веће од 96, као блокови за делимични кружни померај.

Како би се оптимизовала утрошена количина хардверских ресурса, ротатор за $Z \leq 96$ је пројектован као двостепени, слично реализацији предложеној у [126]. Ова реализација омогућава кружни померај за величину циркуланта која није прост број, тј. која се може записати као производ два броја: $Z = g \times Z_g$. Тада је мрежу могуће реализовати коришћењем Z_g g-улазних ротатора чији се излази доводе на улазе g Z_g-улазних ротатора.

Велики број кодова из 5G NR стандарда има циркуланте који омогућавају вишестепено ротирање, јер им је величина одређена са $Z = a \cdot 2^j$, што је описано у одељку 2.2.6.3 и илустровано сликом 3.7. Посматрајући све могуће вредности за Z, закључује се да највећа вредност параметра Z_g мора бити већа од или једнака максималној вредности параметра a, тј. 15. За $a = 15$, максимална вредност величине циркуланта мања од 96 је 60, па у том случају g мора бити $g = 4$. Дакле, за $Z = 60$, први степен ротатора треба да подржи $g = 4$, а други $Z_g = 15$. У случају да је $Z = 30$, први степен треба да подржи $g = 2$, док у случају када је $Z = 15$, први степен само пропушта вредности са улаза. Како би све кружне помераје радила иста мрежа, неопходно је да први степен ротатора буде флексибилан тако да може да ротира један вектор дужине четири, два вектора дужине два или да пропусти вредности са улаза, што је одређено одговарајућом вредношћу параметра g. Како би се за све вредности $Z \leq 96$ задржала иста мрежа за први степен, потребно је да други степен ротатора може кружно да помера све векторе дужине мање или једнаке 24.

a \ j	0	1	2	3	4	5	6	7
2	2	4	8	16	32	64	128	256
3	3	6	12	24	48	96	192	384
5	5	10	20	40	80	160	320	
7	7	14	28	56	112	224		
9	9	18	36	72	144	288		
11	11	22	44	88	176	352		
13	13	26	52	104	208			
15	15	30	60	120	240			

$Z = a \cdot 2^j$

Слика 3.7. Величине циркуланата подржане контролним матрицама 5G NR LDPC кодова

Ако би се на овај начин пројектовао ротатор који подржава све величине циркуланата ($Z \leq 384$), тада би ротатори у првом степену били комплекснији, тј. требало би да раде у једном од пет различитих режима: 1) као један ротатор вектора дужине 16, 2) као два ротатора вектора дужине осам, 3) као четири ротатора вектора дужине четири, 4) као осам ротатора вектора дужине два или 5) тако да само пропусти податке са улаза. Тако је и имплементирано у декодеру презентованом у поглављу 4 и у хардверским реализацијама кодера које су коришћене за поређење са предложеном.

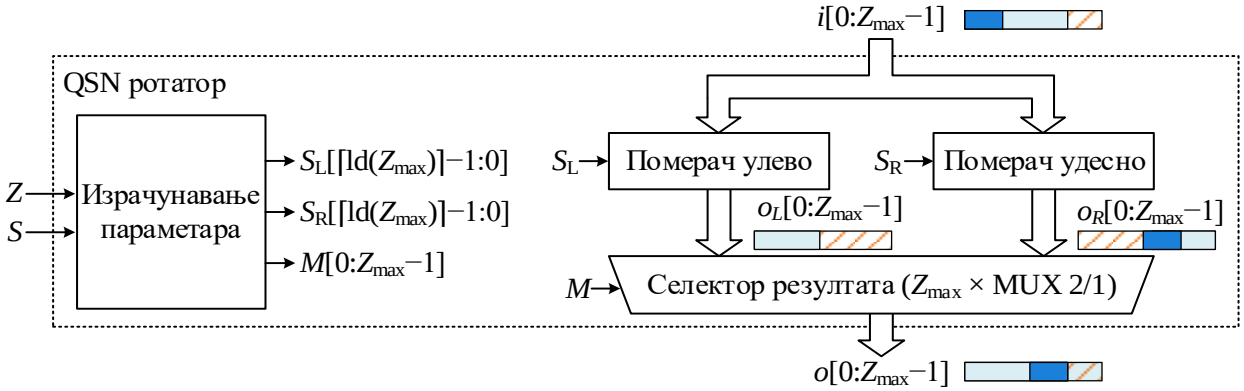
Реализација ротатора чија величина улазног вектора није фиксна није тривијалан задатак. Потребно је обезбедити све могуће кружне помераје за сваку величину улазног вектора, при чему подаци треба да буду груписани у нижим битима. Једно такво решење презентовано је у раду [127], а коришћено је и у дисертацији за реализацију ротатора за све величине улазних вектора мање или једнаке 24. Архитектура ротатора је приказана на слици 3.8. Састоји се од две мреже за линеарни померај од којих једна помера улазни вектор за S_{L} позиција улево, а друга за S_{R} позиција удесно. Ако је кружни померај улево, онда је $S_{\mathrm{L}} = S$, где је са S означен број позиција кружног помераја. Вредност десног помераја је тада одређена са

$$S_{\mathrm{R}} = Z - S_{\mathrm{L}}. \tag{78}$$

Излази ротатора се одређују на следећи начин:

$$o[k] = \begin{cases} o_{\mathrm{L}}[k], & \text{за } 0 \leq k < Z - S \\ o_{\mathrm{R}}[k], & \text{за } Z - S \leq k < Z \end{cases}. \tag{79}$$

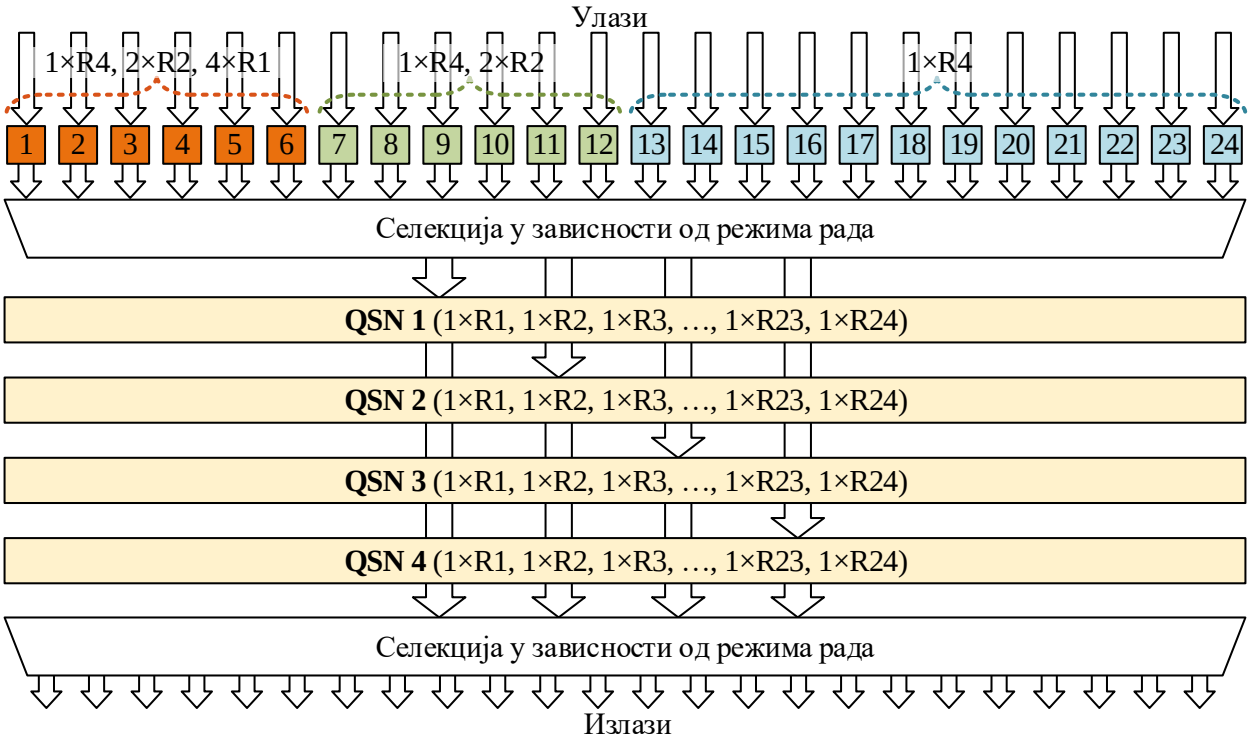
Описана архитектура се у литератури обележава скраћеницом QSN (од енгл. *Quasi-Cyclic LDPC Shifting Network*).



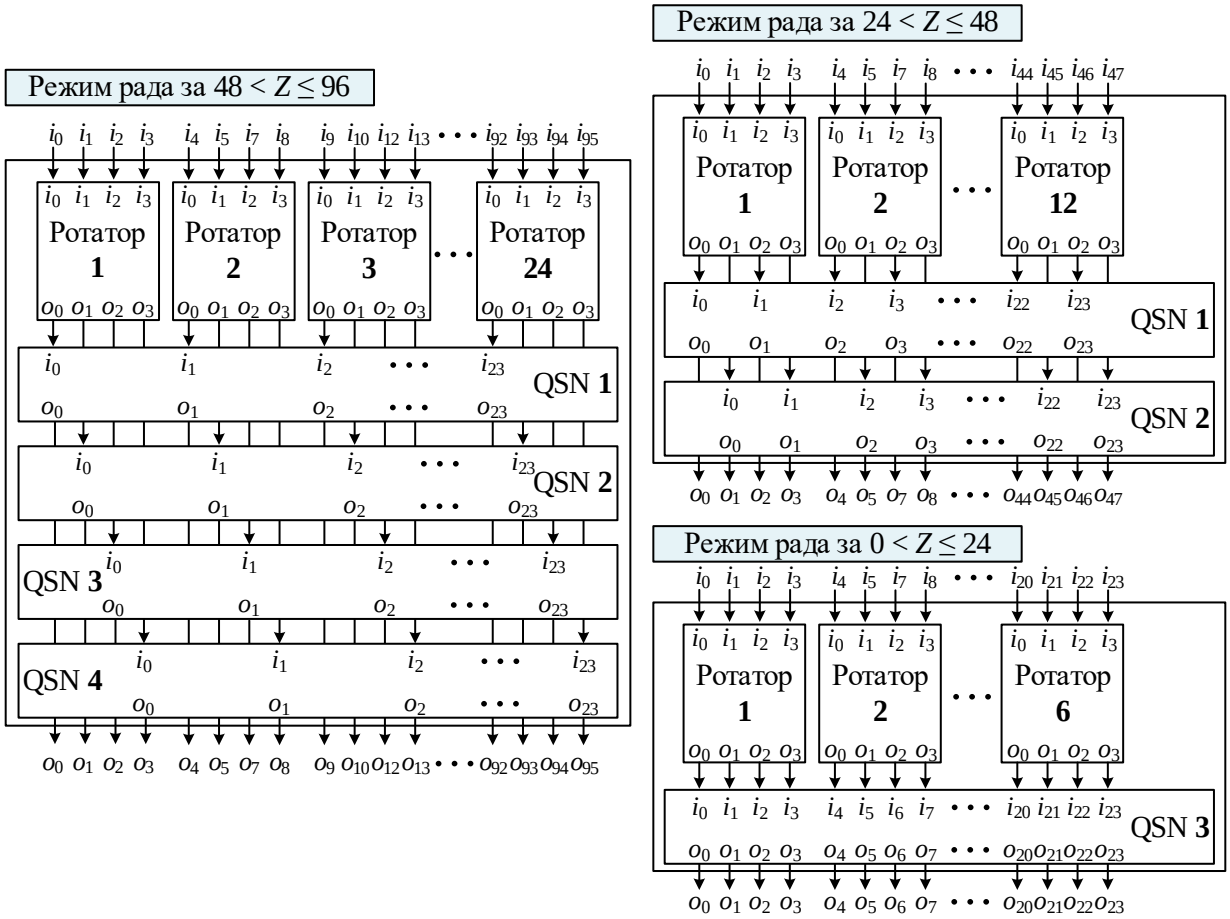
Слика 3.8. Архитектура флексибилне мреже за кружни померај квазицикличних кодова (QSN) [127]

На слици 3.9 приказана је архитектура целог ротатора за $Z \leq 96$ који користи претходно описане ротаторе за векторе величине 1, 2 и 4 као први степен, а QSN ротаторе као други степен. На слици је са $r \times \mathrm{Rz}$ обележен режим рада који ротатор подржава, где r означава број улазних вектора, а z њихову величину. С обзиром на то да је режим 2×R2 потребан само за случајеве када је $24 < Z \leq 48$, а режим $4 \times \mathrm{R1}$ за случајеве када је $Z \leq 24$, нема потребе да сви ротатори у првом степену подржавају сва три раније наведена режима. Због тога су ротатори обележени различитим бојама реализовани тако да подржавају само оне режиме који су неопходни, што додатно штеди ресурсе.

На слици 3.10 приказане су све три могуће конфигурације ротатора са слике 3.9, како би се јасније стекао увид у начин повезивања у различитим режимима рада. У ситуацији када је $Z \leq 24$, користи се трећи QSN ротатор уместо првог. На овај начин се скраћује критична



Слика 3.9. Архитектура ротатора за $Z \leq 96$



Слика 3.10. Различити режими рада ротатора за $Z \leq 96$

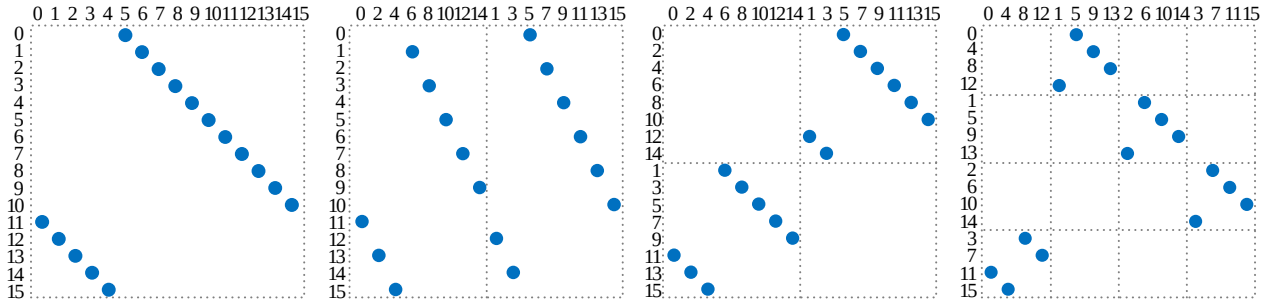
путања од улаза до излаза. Овако се на улазима блокова QSN 1, QSN 2 и QSN 3 налазе само мултиплексери са два улаза (2 у 1), а на улазе QSN 4 се излази из првог степена доводе директно. Коришћење ротатора QSN 1 и у режиму за $Z \le 24$ захтевало би увођење још једног степена мултиплексирања на његовом улазу. Ова наизглед мала измена је од великог значаја за ротатор који је коришћен касније у декодеру (за $Z \le 384$), јер се тада, уместо 4-улазних ротатора у првом степену, користе 16-улазни ротатори са више режима рада, па би за сваки режим рада било потребно увести додатни степен мултиплексирања. Ако се за сваки режим код кога је $Z \le 192$ користе различити QSN ротатори, мултиплексери на улазу остају само мултиплексери 2 у 1. Треба напоменути да се излази свакако мултиплексирају и да нема разлике у критичној путањи од излаза QSN ротатора до излаза целог ротатора.

Правило по коме се одређују вредности помераја за појединачне ротаторе, на основу помераја целог ротатора S, описано је у радовима [126] и [128]. За одређену вредност параметра g, потребно је пронаћи такве S_{f} и S_{c} да важи $S = S_{\mathrm{c}} \cdot g + S_{\mathrm{f}}$, при чему је $0 \le S_{\mathrm{f}} \le g$. Помераји свих ротатора у првом степену се тада постављају на S_{f}. Вредности помераја свих ротатора другог степена су S_{c}, изузев ротатора са индексима $k \ge g - S_{\mathrm{f}} + 1$ за које се помераји постављају на вредност $S_{\mathrm{c}} + 1$.

Када су доступни ротатори за $Z \le 96$, флексибилни ротатор за хардверску имплементацију кодера може се реализовати увођењем додатних пермутација улазног вектора испред и иза четири независна ротатора. Правило по коме се врше наведене пермутације изведено је на основу чињенице да је циркулант величине $Z \times Z$ могуће поделити на мање циркуланте величине $(Z/D) \times (Z/D)$, где је D природан број, ако се изврше одговарајуће пермутације врста и колона. Индекси врста и колона пермутоване подматрице рачунају се коришћењем следећег израза:

$$idx_{\mathrm{new}} = \left(idx_{\mathrm{old}} \bmod D\right) \cdot \frac{Z}{D} + \left\lfloor \frac{idx_{\mathrm{old}}}{D} \right\rfloor, \tag{80}$$

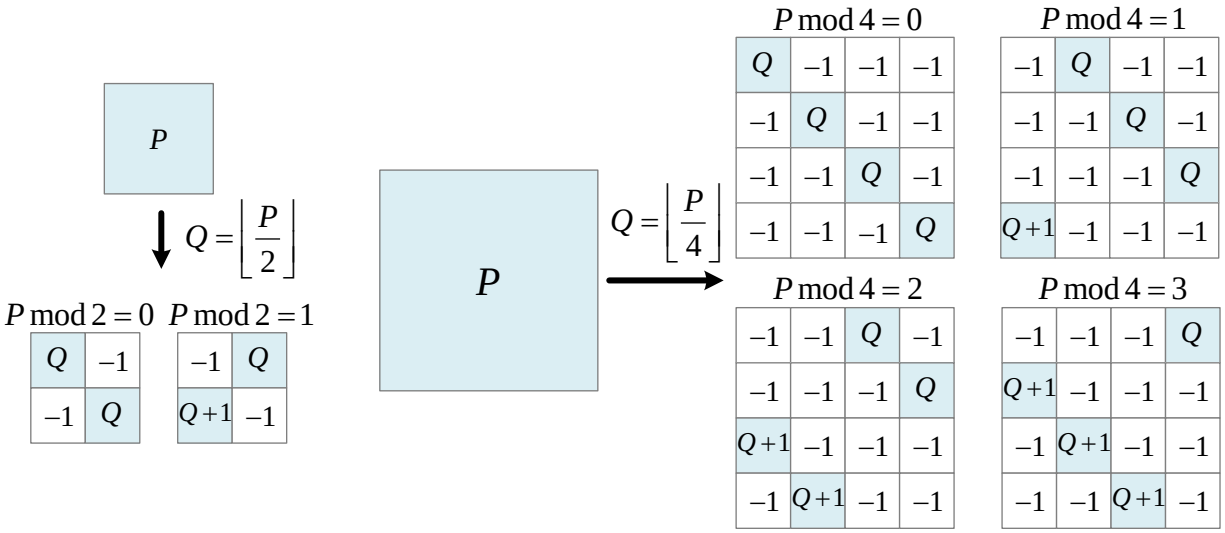
где је са idx_{new} обележен индекс врсте/колоне оригиналне матрице у пермутованој подматрици, а са idx_{old} индекс врсте/колоне у оригиналној подматрици. На слици 3.11 приказани су примери пермутација за $D = 2$ и за $D = 4$. Ради једноставнијег приказа, у примеру су представљене пермутације оригиналног циркуланта величине $Z = 16$, али је принцип потпуно исти за било коју величину циркуланта. Из примера се јасно види да се у пермутованој подматрици могу идентификовати независни циркуланти мање величине.



Слика 3.11. Примери дељења једног циркуланта на више мањих: (а) оригинални циркулант, (б) оригинални циркулант са пермутованим колонама за D = 2, (в) финални пермутовани циркулант за D = 2, и (г) финални пермутовани циркулант за D = 4 [58]

На слици 3.12 приказани су резултати описаних пермутација у општем случају. За једну вредност из пермутационе матрице, тј. од једног циркуланта, добијају се четири нове

вредности у новој матрици за $D=2$, а шеснаест нових вредности за $D=4$. Неке од њих представљају нове кружно померене јединичне матрице, а неке нула-матрице.



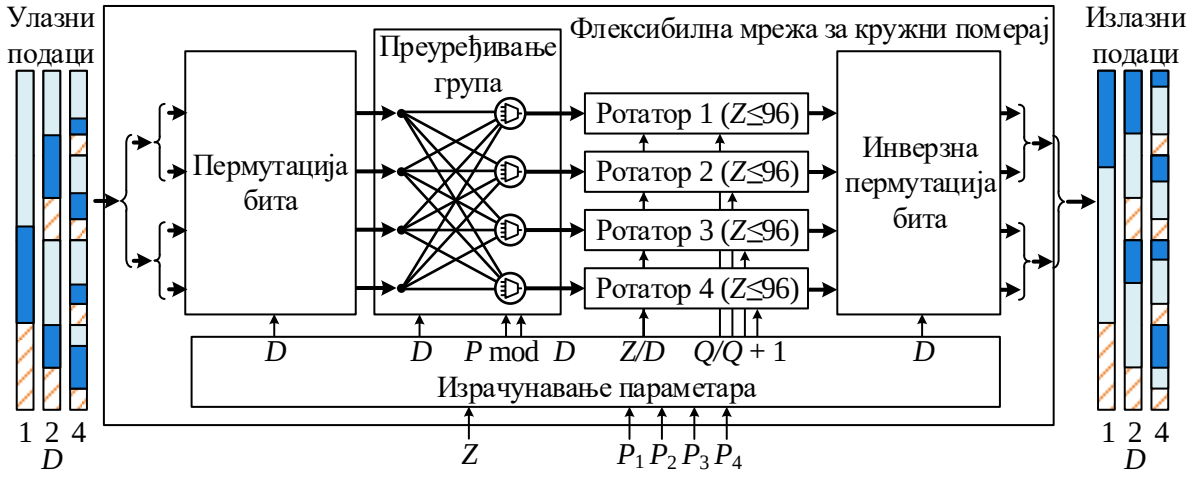
Слика 3.12. Резултати дељења циркуланта пермутацијама у општем случају за D = 2 и за D = 4. Параметар P представља вредност кружног помераја из пермутационе матрице. Вредност −1 представља нула-матрицу. [58]

Подела једног великог циркуланта на више мањих коришћена је у ранијим радовима за дизајн LDPC декодера који имају мањи број паралелних процесорских елемената и за спречавање хазарда услед превременог читања [48]. У овој дисертацији је иста идеја примењена на пројектовање флексибилне мреже за кружни померај с обзиром на то да је показано да се кружни померај великог вектора може поделити на више кружних помераја његових делова. Испред четири паралелна ротатора за $Z \le 96$ потребно је поставити комбинациону мрежу која ће пермутовати улазне податке тако да они буду у распореду одређеном изразом (80). У случају да је мрежа за кружни померај конфигурисана да ротира четири вектора мале дужине, није потребно радити пермутације (тј. $D=1$). У случају да се ротирају два вектора средње дужине (тј. за $96 < Z \le 192$) или један вектор велике дужине ($192 < Z \le 384$), параметар D треба поставити на вредност два или четири респективно. На тај начин се добија више независних група података које се могу ротирати у паралелно постављеним ротаторима. Излази ротатора су такође груписани, па је потребно урадити инверзну пермутацију како би се подаци дугачког вектора преуредили у исправан редослед.

Архитектура мреже за кружни померај која користи описани метод приказана је на слици 3.13. Може се уочити још један блок на путањи података поред горенаведених блокова за пермутације и ротирање. Овај блок служи за преуређивање редоследа група на излазу из блока за улазну пермутацију бита и неопходан је јер циркуланти у пермутованој подматрици нису увек постављени по главној дијагонали. У зависности од остатка при дељењу вредности помераја са параметром D, потребно је изменити редослед група тако да се након инверзне пермутације добију бити у исправном редоследу. На пример, ако се посматра случај за $D=4$ и остатак при дељењу $P \bmod 4 = 1$, са слике 3.12 се може закључити да је потребно прву групу поставити као последњу, другу као прву, трећу као другу и четврту као трећу. Слично, за $P \bmod 4 = 2$, прва група треба да се постави на треће место, друга група на четврто, трећа група на прво и четврта на друго итд.

Описана архитектура омогућава да су сви излазни бити вектора, дужине различите од 96, 192 или 384, груписани на једној страни блока података, тј. између њих се не појављују неки невалидни подаци. Неколико примера је приказано на слици 3.13. Плави објекти представљају валидне податке које је потребно ротирати. Усвојени су примери за $Z \le 96$,

$96 < Z \le 192$ и $192 < Z \le 384$. Невалидни подаци су означени шрафираном површином. Ова особина мреже за кружни померај је веома важна јер у многоме омогућава једноставнији дизајн неких других компоненти кодера, као што су блок за израчунавање основних контролних бита, улазни интерфејс за податке и излазни интерфејс за податке.



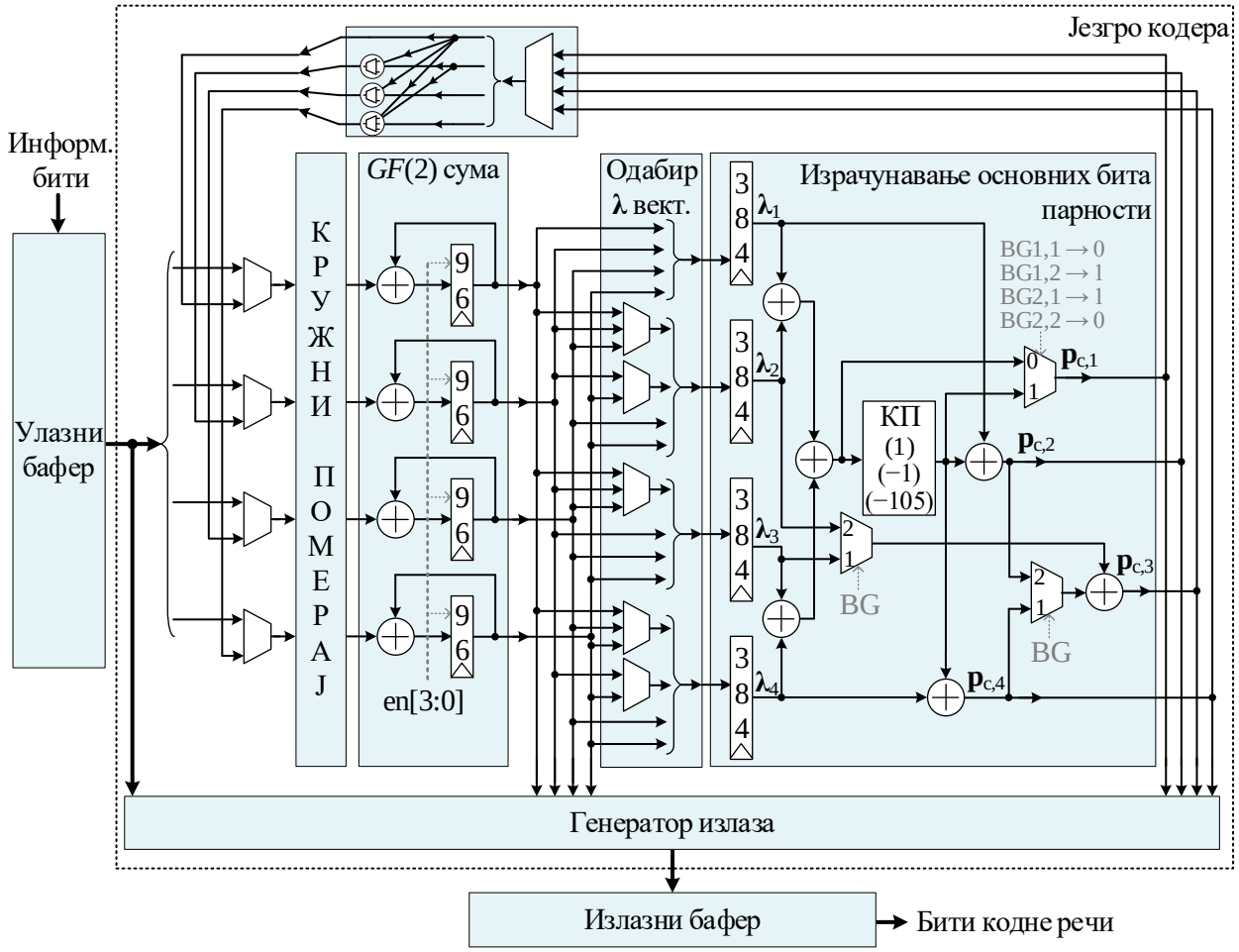
Слика 3.13. Архитектура флексибилне мреже за кружни померај [58]

3.3.2. Архитектура кодера

Хардверска архитектура предложеног флексибилног кодера за 5G NR стандард приказана је на слици 3.14. Улазни информациони бити се чувају у улазном баферу реализованом помоћу двопортне меморије у коју се може сместити највише $2k_{\max} = 16896$ информационих бита. Меморија се користи на начин који омогућава двоструко баферисање улаза. Једна половина локација се користи за читање већ уписаних информационих бита током кодовања текуће кодне речи, док се друга половина користи за упис информационих бита нове кодне речи са улаза. Након што се заврши кодовање једне речи, у улазном баферу су на другим меморијским локацијама одмах доступни информациони бити наредне речи, па због тога не постоји потреба за додатном иницијализацијом кодера која би довела до чекања и смањене брзине рада.

У зависности од величине циркуланта, кодер може радити у једном од три режима. Ако је величина циркуланта у опсегу $192 < Z \le 384$, кодер серијски процесира један циркулант по циклусу такта. Из улазног бафера се у сваком такту чита по једна група информационих бита максималне дужине 384. Прочитани информациони бити се затим кружно померају и сабирају у коначном пољу $GF(2)$ како би се добиле вредности међурезултата ($\boldsymbol{\lambda}$ вектора). Сва сабирања у $GF(2)$ су реализована помоћу „ексили" кола. Ако је величина циркуланта у опсегу $96 < Z \le 192$, кодер серијски процесира највише два циркуланта у једном циклусу такта, а који припадају врстама пермутационе матрице одабраним за заједничко процесирање, при раније описаној оптимизацији редоследа кодовања. У овом случају, вектор бита који се прочита из улазног бафера састоји се од групе информационих бита (чија је максимална дужина 192) и копије истих бита смештене на вишим позицијама унутар речи. Копирање информационих бита обезбеђује блок улазног интерфејса који је посебно пројектован и није приказан на слици 3.14. Мрежа за кружни померај се тада конфигурише тако да ради као два независна ротатора за улазне векторе чија је дужина одређена величином циркуланта. Коначно, трећи режим рада настаје у случају да је величина циркуланта у опсегу $2 < Z \le 96$, када кодер серијски процесира максимално четири циркуланта из четири суседне колоне дефинисане оптимизованим редоследом кодовања.

Слично као у другом режиму рада, у истим меморијским локацијама улазног бафера налазе се четири идентичне копије групе информационих бита, за шта је такође одговоран блок улазног интерфејса. Свака од наведене четири групе информационих бита ротира се независно у ротаторима, а кружно померене вредности се користе за израчунавање четири независне суме у $GF(2)$. Током израчунавања међурезултата, сви информациони бити са улаза у мрежу за кружни померај прослеђују се генератору излаза, који их смешта у излазни бафер.



Слика 3.14. Хардверска архитектура предложеног LDPC кодера за 5G NR стандард [58]

Након што се заврши процесирање свих циркуланата из прве четири врсте, међурезултати су спремни за даљу обраду и израчунавање основних контролних бита. С обзиром на то да $\boldsymbol{\lambda}$ вектори могу имати различите дужине и да се израчунавају у различитим временским тренуцима, зависно од режима рада кодера, потребно је преуредити их тако да претходне разлике не утичу на архитектуру блока за израчунавање основних контролних бита. За то је одговоран блок означен са „Одабир $\boldsymbol{\lambda}$ вектора“ на слици 3.14.

Све врсте пермутационе матрице након четврте процесирају се према оптимално одређеном редоследу на исти начин као и прве четири врсте. Током учитавања информационих бита за врсте из којих се добијају додатни контролни бити, основни контролни бити се израчунавају у паралели, на основу претходно израчунатих међурезултата из прве четири врсте. Као што је описано у потпоглављу 3.1, израчунавање основних контролних бита разликује се за кодове изведене из основног графа 1 и за кодове изведене из основног графа 2, као и за различите вредности величине циркуланата. Међутим, већина логичких ресурса („ексили” кола) може се искористити за израчунавање у сва четири случаја

ако се на путањи података постави неколико мултиплексера. Ротатор који је на слици 3.14 означен са КП(1)(−1)(−105) врши кружни померај улазног вектора за једну позицију улево, једну позицију удесно или за 105 позиција удесно, у зависности од примењеног кода. С обзиром на то је овим ротатором потребно реализовати свега три вредности кружног помераја, хардверски ресурси које заузима неупоредиво су мањи од ресурса које заузима главна мрежа за кружни померај, иако је за помераје (1) и (−1) потребно подржати све величине циркуланта. Основни контролни бити се такође прослеђују генератору излаза, који их уписује у меморију према одговарајућем редоследу.

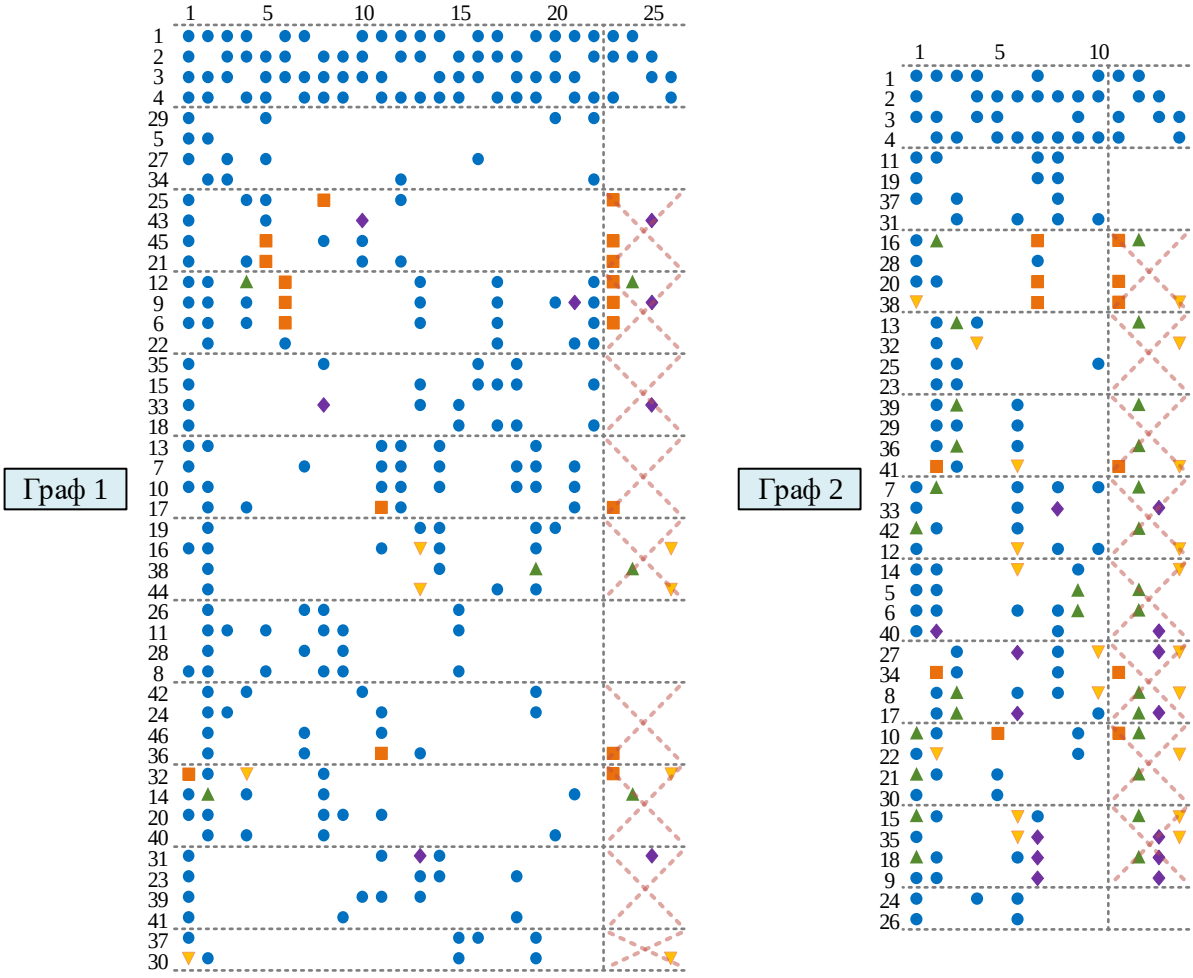
Израчунати основни контролни бити су потребни и за израчунавање додатних контролних бита, па се и они, поред информационих бита, доводе на улазе мреже за кружни померај. У зависности од позиције циркуланта, селектује се једна од четири групе основних контролних бита. Одабрани контролни бити се, у случају да је $Z \leq 192$, копирају на исти начин као што је то урађено за групе информационих бита на улазу. Које ће групе бита, информационе или контролне, бити прослеђене на улазе мреже за кружни померај, одређују четири независне групе мултиплексера контролисане на основу података о циркулантима који се тренутно процесирају. Излази регистара у којима се чува сума у $GF(2)$ сада су додатни контролни бити који се такође прослеђују генератору излаза.

Како би се обезбедила висока учестаност сигнала такта, примењена је техника проточне обраде. Сходно томе, путања података је издељена великим бројем регистара, како би се смањила критична логичка путања. На пример, само мрежа за кружни померај има четири степена проточне обраде. Стога се основни контролни бити не могу израчунати непосредно пре почетка процесирања циркуланата којим се израчунавају додатни контролни бити. Из наведеног разлога, редослед кодовања мора да обезбеди да не постоје конфликти услед превременог читања, тј. не сме се дозволити да се користи садржај регистара у којима се чувају основни контролни бити пре него што се у њих упишу валидни подаци. Како би се омогућило довољно времена за избегавање наведених конфликата, идеално би било да прве врсте које се користе за израчунавање додатних контролних бита уопште немају циркуланте у колонама које одговарају основним контролним битима. Ово је обезбеђено постављањем додатног ограничења у процесу оптимизације редоследа кодовања.

3.4. РЕЗУЛТАТИ И ДИСКУСИЈА

3.4.1. РЕЗУЛТАТИ ОПТИМИЗАЦИЈЕ РЕДОСЛЕДА ПРОЦЕСИРАЊА

Оптимизацијом редоследа процесирања остварено је веома значајно смањење времена кодовања за кодове кратке и средње дужине. Примери оптимизованих редоследа процесирања врста за кодове изведене из оба графа приказани су на слици 3.15. Кружићима су представљене позиције циркуланата који одређују кружни померај информационих бита, док остали знаци представљају циркуланте који одређују кружни померај основних бита парности. Бројеви представљају индекс врсте или колоне. Лако је уочити да су врсте груписане тако да је велики број циркуланата суседних врста у истој колони. Такође, процесирање свих циркуланата из колона које одговарају битима парности могуће је спојити са процесирањем неких циркуланата из информационих колона, што је од великог значаја за додатно смањење времена кодовања.



Слика 3.15. Распоред циркуланата у оптимизованом редоследу процесирања врста за 5G NR кодове изведене из оба основна графа за случај када су на располагању четири независна ротатора [58]

Свеукупни резултати оптимизационог поступка приказани су у табели 3.2. Са N_{CPC} обележен је укупан број циклуса такта потребних да би се завршило кодовање једне кодне речи, а са $N_{\mathrm{CPC,opt}}$ је означен број циклуса такта за оптимизовани редослед процесирања, док су са $I_{\mathrm{opt/unopt}}$ и $I_{\mathrm{opt/serial}}$ обележена убрзања у процентима постигнута оптимизованим процесирањем када се оно пореди са неоптимизованим редоследом и ако се пореди са конвенционалним серијским процесирањем, респективно. Убрзања се могу израчунати коришћењем следећих израза:

$$I_{\mathrm{opt/unopt}} = \frac{N_{\mathrm{CPC}} - N_{\mathrm{CPC,opt}}}{N_{\mathrm{CPC,opt}}} \cdot 100\%, \tag{81}$$

$$I_{\mathrm{opt/serial}} = \frac{N_{\mathrm{CPC,serial}} - N_{\mathrm{CPC,opt}}}{N_{\mathrm{CPC,opt}}} \cdot 100\%, \tag{82}$$

где је са $N_{\mathrm{CPC,serial}}$ означен број тактова потребних за серијско кодовање (265 за граф 1 и 150 за граф 2).

Резултати приказани у табели 3.2 показују да се, коришћењем више од једног ротатора за кодовање и оптимизацијом редоследа процесирања, постиже убрзање од просечно 1,68 пута, ако су на располагању два ротатора (за кодове код којих је $96 < Z \leq 192$), и убрзање од просечно 2,65 пута, ако су на располагању 4 ротатора (за кодове код којих је $Z \leq 96$). Може

се приметити да је остварено убрзање веће за кодове изведене из основног графа 2. Ово је последица тога што је за те кодове процентуално знатно већи број циркуланата у колонама које одговарају основним битима парности, него за кодове изведене из основног графа 1, а процесирање свих њих је спојено са процесирањем неких циркуланата из информационих колона.

Табела 3.2. Резултати оптимизације редоследа процесирања генетичким алгоритмом [58]

Редослед процесирања	N_{CPC}		$N_{\mathrm{CPC,opt}}$		$I_{\mathrm{opt/unopt}}(\%)$		$I_{\mathrm{opt/serial}}(\%)$	
	BG1	BG2	BG1	BG2	BG1	BG2	BG1	BG2
Серијски	265	150	-	-	-	-	0	0
Серијски са 2 ротатора	223	136	165	86	35,15	58,14	60,61	74,42
Серијски са 4 ротатора	180	94	107	53	68,22	77,35	147,66	183,02

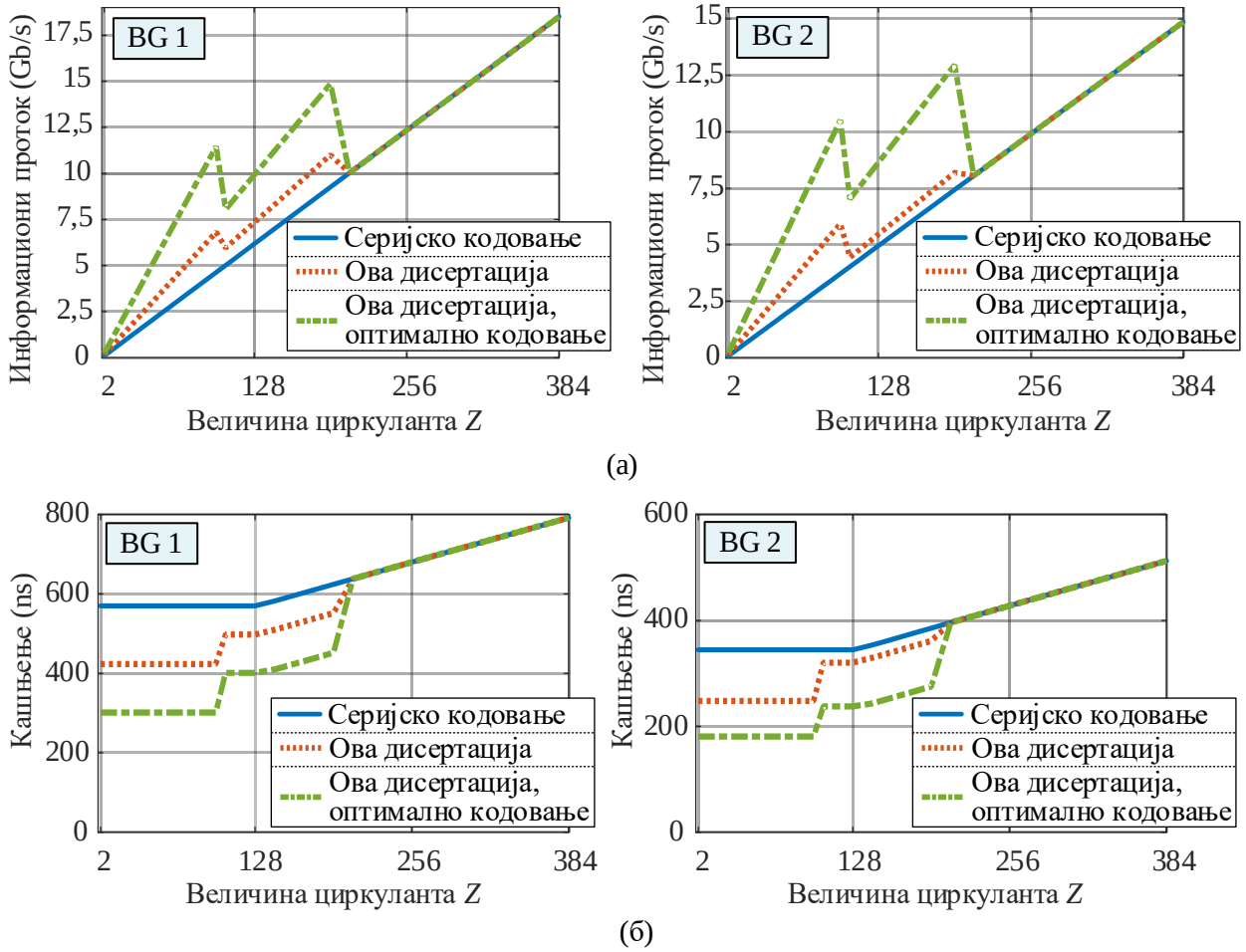
3.4.2. ПРОТОК, КАШЊЕЊЕ И ЕФИКАСНОСТ ИСКОРИШЋЕЊА ХАРДВЕРСКИХ РЕСУРСА

Предложена архитектура кодера имплементирана је на развојном систему Xilinx ZCU111 са Zynq Ultrascale+ RF-SoC чипом (XCZU28DR). За синтезу и имплементацију свих блокова реализованих у дисертацији коришћен је софтвер Xilinx Vivado Design Suite 2019.2 Остварена максимална учестаност сигнала такта кодера је $f_{\max} = 580$ MHz, док максимални информациони проток за кодове изведене из основног графа 1 износи 18,51 Gb/s, а за граф 2 кодове 14,87 Gb/s. На слици 3.16.а) приказани су детаљнији резултати, тј. остварени информациони проток за све величине циркуланата и поређење са оригиналним серијским процесирањем. Може се приметити значајно побољшање у протоку за кодове чија је величина циркуланата мања или једнака 192, а посебно у случају када је редослед процесирања оптимизован генетичким алгоритмом. Велике вредности информационог протока остварене су за много краће кодне речи него што је то случај са уобичајеним серијским процесирањем.

Кашњење кодера за различите величине циркуланата приказано је на слици 3.16.б). Одређено је као време које протекне између последњег трансфера информационих бита једне кодне речи на улазу кодера и последњег трансфера кодних бита исте кодне речи на излазу кодера. Остварено кашњење је мање од 1 μs за све кодове, а из резултата се може закључити да предложена архитектура остварује значајно мања кашњења него серијски кодер. Треба нагласити да кашњење кодера, поред самог трајања кодовања, зависи и од протока излазног интерфејса. Број бита излазног податка потребно је подесити тако да проток излазног интерфејса може да подржи максимални кодни проток језгра кодера. Најнижи кодни количник за кодове изведене из графа 1 је $R_{\mathrm{BG1,min}} = 1/3$, а из графа 2 $R_{\mathrm{BG2,min}} = 1/5$, што даје максималне кодне протоке $T_{\mathrm{cod,max,BG1}} = 18{,}51\ \mathrm{Gb/s} / R_{\mathrm{BG1,min}} = 55{,}53\ \mathrm{Gb/s}$ за граф 1 и $T_{\mathrm{cod,max,BG2}} = 14{,}87\ \mathrm{Gb/s} / R_{\mathrm{BG2,min}} = 74{,}35\ \mathrm{Gb/s}$ за граф 2. На основу тога је потребни проток интерфејса 74,35 Gb/s, што се може остварити ширином податка од 128 бита на такту учестаности 580 MHz. Излазни интерфејс у једном такту из излазног бафера може да прочита групе кодних бита дужине *Z*. Због тога је кашњење кодера (енгл. *encoder latency*) за све $Z \le 128$ одређено бројем циклуса такта потребних за кодовање и бројем циклуса такта потребних да се из излазног бафера прочита $n_{\mathrm{b}} - 2$ циркуланата (овде се ради и пунктирање првих $2Z$ бита):

$$l_{\mathrm{e},Z\le 128} = \left(N_{\mathrm{CPC}} + n_{\mathrm{b}} - 2\right)\cdot\frac{1}{f_{\mathrm{CLK}}}. \tag{83}$$

За дуже кодне речи, када је величина циркуланта већа од ширине интерфејса, кашњење је веће јер се мора сачекати слање свих података пре него што се прочита нова група бита из излазног бафера. Ово понашање се јасно види на дијаграмима са слике 3.16.б).



Слика 3.16. Остварене перформансе имплементираног 5G NR LDPC кодера на FPGA чипу за три различита начина процесирања контролне матрице: (а) информациони проток и (б) кашњење [58]

С обзиром на то да је главни циљ дисертације постизање велике ефикасности искоришћења хардверских ресурса, преостаје још анализа у том погледу. Како би се упоредили резултати добијени предложеном архитектуром са резултатима које би дала друга архитектурална решења присутна у литератури, на истој хардверској платформи имплементирана су три таква решења. Ова решења су у алгоритамском погледу (али и делимично у погледу реализације) описана у одељку 3.2.1. Имплементирани су: 1) серијски кодер без подржаног додатног паралелизма за краће кодове, 2) кодер који реализује по колонама паралелно кодовање (метод коришћен у [105]), и 3) кодер који реализује по врстама паралелно кодовање (метод коришћен у [103] и [106]).

Серијски кодер суштински има исту архитектуру као и предложени кодер представљен у одељку 3.3.2. Међутим, све редундантне хардверске компоненте су уклоњене како би се стекла јаснија слика о количини потребних ресурса за реализацију. Мрежа за кружни померај је реализована тако да нема више режима рада већ је у стању да врши ротирање једног улазног вектора за све $Z \le 384$. Блок за одабир међурезултата ($\boldsymbol{\lambda}$ вектора) и блок за копирање основних бита парности су потпуно уклоњени. Поред тога, потребно је чувати

мање података о контролним матрицама кодова с обзиром на то да не постоји потреба за меморисањем оптимизованих редоследа процесирања.

Као што је поменуто у одељку 3.2.1, по колонама паралелна архитектура је раније коришћена за хардверску реализацију кодера за Wi-Fi стандард [105]. Како би се урадило коректно поређење, овде је имплементиран по колонама паралелни кодер за 5G NR стандард који подржава све специфициране кодове. С обзиром на то да се сви циркуланти једне колоне процесирају паралелно, за реализацију овог кодера је потребно 46 ротатора за све $Z \leq 384$ и исти број блокова за израчунавање $GF(2)$ сума. Ротатори су реализовани као и ротатор за серијски кодер. С обзиром на то да је примењена проточна обрада како би се повећала учестаност такта, потребно је направити паузу од два циклуса такта пре него што се израчунају основни контролни бити, па је укупан минимални број циклуса такта 28 за кодовање кодова изведених из графа 1 и 16 за кодове изведене из графа 2. Основни бити парности се израчунавају коришћењем истог блока као у предложеном кодеру. Подаци о контролним матрицама су смештени у блок RAM (енгл. *Block Random Access Memory, BRAM*) меморије, у форми осам пермутационих матрица задатих стандардом за највеће величине циркуланата. Вредности кружног помераја за мање циркуланте израчунавају се у комбинационој логици. Узимајући у обзир то да је неопходно паралелно прочитати 46 вредности кружног помераја, за чување матрица је потребно више паралелно постављених RAM меморија.

По врстама паралелна архитектура раније је примењена за кодовање 5G NR кодова [106], па је сличан приступ реализован и за поређење са архитектуром предложеном у дисертацији. Како је потребно паралелно процесирати све циркуланте из једне врсте, потребно је употребити 26 ротатора за све $Z \leq 384$. Излази ротатора се сумирају паралелно у комбинационој мрежи реализованој као стабло „ексили" логичких кола. Као и код осталих архитектура, подаци о контролним матрицама, тј. пермутационе матрице за највеће циркуланте, чувају се у блок RAM меморијама. Међутим, пошто је број кружних помераја који је у паралели потребан мањи него код по колонама паралелне архитектуре, то је и број меморија мањи.

Резултати имплементације кодера предложеног у дисертацији и поређење са горенаведеним архитектурама приказани су у табели 3.3. Наведени ресурси су ресурси које заузимају језгра кодера, без улазних и излазних бафера. Стога се блок RAM меморије наведене у табели користе само за чување параметара контролних матрица. У зависности од усвојене архитектуре, добијене су различите максималне учестаности сигнала такта. Иако је и код њих примењена техника проточне обраде са великим бројем степени, архитектуре које користе велики паралелизам могу да раде на нижој учестаности такта због загушења које настаје при рутирању и због велике дужине линија за повезивање. Међутим, како би се направило боље поређење са становишта архитектуре, добијени информациони протоци се могу нормализовати дељењем са учестаношћу сигнала такта:

$$T_{\text{norm}} = \frac{T}{f_{\text{CLK}}}. \tag{84}$$

Нормализоване вредности протока израчунате су за кодове изведене из оба основна графа и за све величине циркуланата. У табели 3.3 приказана је њихова просечна вредност. Ефикасност искоришћења хардверских ресурса је израчуната као проток (нормализовани или остварени) подељен са бројем употребљених компоненти FPGA чипа. Посматрани су укупан број употребљених лукап табела (за Ultrascale+ фамилију лукап табеле су шестоулазне), број флип-флопова, број употребљених блок RAM меморија и број Slice блокова. Један Slice блок у Ultrascale+ фамилији садржи укупно осам шестоулазних лукап табела и шеснаест флип-флопова. Иако је анализа броја употребљених Slice блокова редундантна, поред анализе броја лукап табела и флип-флопова, она је задржана јер се у пракси често користи за

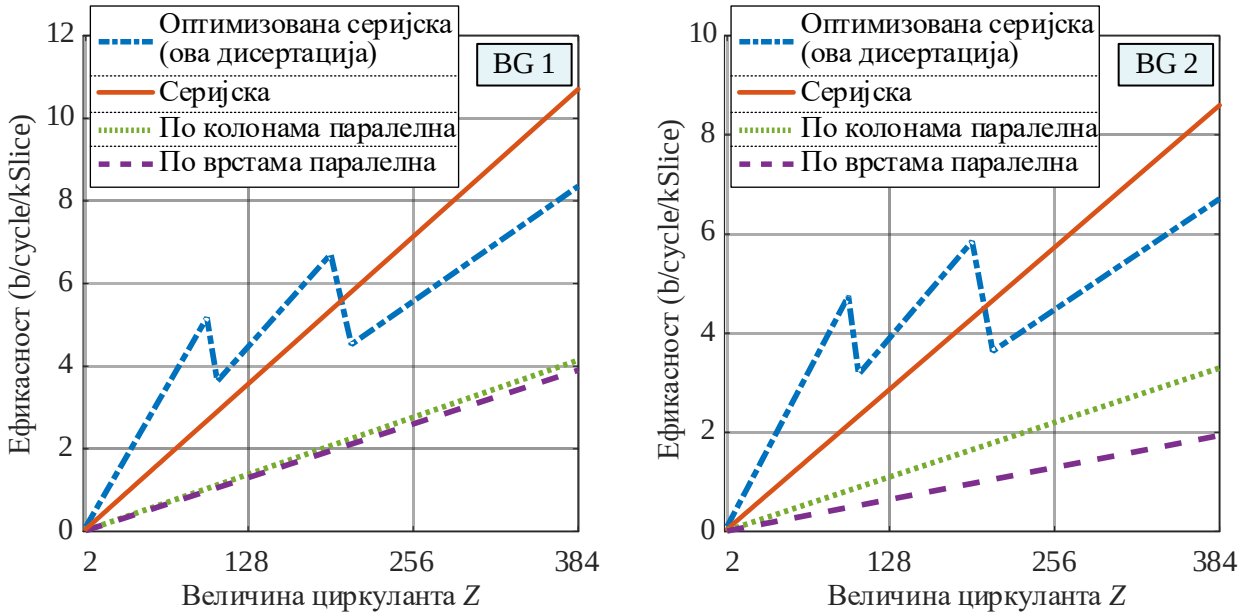
евалуацију заузећа комбинационих и регистарских елемената здружено. Резултати приказани у табели показују да предложена архитектура даје убедљиво највећу ефикасност искоришћења хардверских ресурса.

Табела 3.3. Резултати имплементације различитих архитектура флексибилног 5G NR LDPC кодера и поређење просечних вредности протока и ефикасности искоришћења хардверских ресурса [58]

Архитектура	Заузеће ресурса (N_{res})				$T_{norm,avg}$ (b/cycle)	f_{max} (GHz)	T_{avg} (Gb/s)	Ефикасност ($T_{(norm),avg}$ / N_{res})			
	N_{Slice}	N_{LUT}	N_{FF}	N_{BRAM}				b/cycle/ kSlice	b/cycle/ kLUT	b/cycle/ kFF	Gb/s/ kSlice
По колонама паралелна [105]	42257	248641	246960	6	**61,9**	330	**20,43**	1,46	0,25	0,25	0,48
По врстама паралелна [106]	26180	126986	131185	3,5	30,1	470	14,15	1,15	0,24	0,23	0,54
Серијска	1729	8712	9975	3	6,6	**580**	3,83	3,82	0,76	0,66	2,22
Оптимизована серијска (ова дисертација)	2215	11156	9635	5	10,3	**580**	5,97	**4,65**	**0,92**	**1,07**	**2,70**

N_{res}: број употребљених компоненти FPGA чипа (N_{Slice} : број Slice блокова, N_{LUT} : број лукап табела, N_{FF} : број флип-флопова, N_{BRAM} : број блок RAM-ова); $T_{norm,avg}$: нормализовани проток усредњен по свим вредностима величине циркуланта у битима по циклусу; T_{avg} : остварени проток усредњен по свим вредностима величине циркуланта; kSlice/kLUT/kFF: хиљаду Slice блокова/лукап табела/флип-флопова

Како би се стекао јаснији увид у резултате ефикасности за различите величине циркуланата, осим просечних вредности датих у табели 3.3, на слици 3.17 приказана је ефикасност искоришћења Slice блокова за кодове изведене из оба графа и за сваку величину циркуланта предвиђену стандардом. Са графика се може закључити да су архитектуре код којих је имплементиран велики паралелизам изузетно неефикасне, када се упореде са серијском архитектуром и оптимизованом архитектуром предложеном у дисертацији.



Слика 3.17. Нормализована ефикасност искоришћења хардверских ресурса (Slice блокова) за различите архитектуре 5G NR LDPC кодера у зависности од величине циркуланта [58]

Коришћењем више паралелних кодера, оптимизованих по ефикасности, могу се остварити протоци и већи од протока изузетно паралелних архитектура уз коришћење значајно мање хардверских ресурса. Ипак, архитектуре које користе већи паралелизам имају мање кашњење. Међутим, с обзиром на то да је, за све дужине кодне речи, кашњење и серијске и оптимизоване серијске архитектуре мање од 1 μs и узимајући у обзир чињеницу да је захтев за кашњење целог комуникационог ланца дефинисан за кориснички сценарио веома поузданих комуникација малог кашњења (енгл. *Ultra-Reliable Low-Latency Communication, URLLC*) чак 1 ms [61], што је 1000 пута веће време, не постоји ни један преостали разлог за коришћење изузетно паралелних архитектура.

3.5. ПРИМЕНА ОСТВАРЕНИХ ДОПРИНОСА НА КОДОВАЊЕ ДРУГИХ LDPC КОДОВА

Већина доприноса описаних у овом поглављу може се применити и у архитектурама за кодовање других LDPC кодова. Описани метод пројектовања мреже за кружни померај, која омогућава независну ротацију два или четири мањих улазних вектора или ротацију једног великог улазног вектора, може се генерализовати. Даљим дељењем подматрице циркуланта може се добити већи број још мањих независних циркуланата. Стога је могуће и пројектовати мрежу за кружни померај која, поред претходно описаних режима рада, ротира и више од четири независна улазна вектора. Такав приступ није примењен за реализацију 5G NR LDPC кодера, јер се комплексност ротатора и осталих делова кодера додатно усложњава, а због изражене нерегуларности 5G NR LDPC кодова, добици у протоку нису тако велики као што је то био случај раније. Додатно, процесирање прве четири врсте у предложеној архитектури не може се додатно паралелизовати. Међутим, друге архитектуре кодера или декодера могу имати више користи од додатне флексибилности мреже за кружни померај. На пример, како би се остварила већа ефикасност искоришћења хардверских ресурса, по врстама паралелне архитектуре би могле да користе ротаторе који кружно померају све информационе бите у паралели само за кодове код којих су величине циркуланата мале, а комбинују паралелно и серијско процесирање за веће величине циркуланата. Ипак, изражена нерегуларност контролних чворова 5G NR LDPC кодова онемогућава искоришћење такве флексибилности у пуном капацитету, док би се за регуларније кодове могло остварити знатно већа ефикасност искоришћења хардверских ресурса. Исти закључци важе и за кодове код којих је регуларност варијабилних чворова већа ако се примењује по колонама паралелна архитектура.

Веома слична архитектура кодера и готово исти принцип кодовања и оптимизације редоследа процесирања може се применити на кодове из WiMAX стандарда. У односу на предложену 5G архитектуру, постоје две најважније разлике. Прво, величине циркуланата WiMAX кодова узимају вредности између 24 и 96 са инкрементима по четири. Стога би мрежу за кружни померај, уместо уз помоћ четири ротатора за $Z \le 96$, требало пројектовати уз помоћ четири ротатора за $Z \le 24$. Поред тога, WiMAX кодови имају само основне бите парности, гледано у номенклатури усвојеној за 5G кодер, али њихов број варира зависно од кодног количника од $4Z$ (за $R = 5/6$) до $12Z$ (за $R = 1/2$). Због тога би, уместо паралелног израчунавања контролних бита, било ефикасније израчунавати их коришћењем рекурзивних израза сличним изразима (37). У даљем тексту је дат предлог једне такве архитектуре.

Структура WiMAX LDPC кодова дата је изразом (14). За све кодове задате стандардом важи да су $\mathbf{Q}_{\mathrm{P}_{1,k_\mathrm{b}+1}}$ и $\mathbf{Q}_{\mathrm{P}_{m_\mathrm{b},k_\mathrm{b}+1}}$ исти циркуланти, па се наведени израз може записати и другачије:

$$\mathbf{H}=\left[\mathbf{H}_1 \mid \mathbf{H}_2\right]=\left[\begin{array}{cccc|c:cccccc} \mathbf{Q}_{1,1} & \mathbf{Q}_{1,2} & \cdots & \mathbf{Q}_{1,k_\mathrm{b}} & \mathbf{I}^{(S_1)} & \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{Q}_{2,1} & \mathbf{Q}_{2,2} & \cdots & \mathbf{Q}_{2,k_\mathrm{b}} & \mathbf{0} & \mathbf{I} & \mathbf{I} & \cdots & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{Q}_{r-1,1} & \mathbf{Q}_{r-1,2} & \cdots & \mathbf{Q}_{r-1,k_\mathrm{b}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{Q}_{r,1} & \mathbf{Q}_{r,2} & \cdots & \mathbf{Q}_{r,k_\mathrm{b}} & \mathbf{I}^{(S_2)} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I} & \mathbf{I} & \cdots & \mathbf{0} \\ \mathbf{Q}_{r+1,1} & \mathbf{Q}_{r+1,2} & \cdots & \mathbf{Q}_{r+1,k_\mathrm{b}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{I} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{Q}_{m_\mathrm{b},1} & \mathbf{Q}_{m_\mathrm{b},2} & \cdots & \mathbf{Q}_{m_\mathrm{b},k_\mathrm{b}} & \mathbf{I}^{(S_1)} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I} \end{array}\right]. \tag{85}$$

Кодовање се такође реализује ротирањем информационих бита и рачунањем њихове суме у $GF(2)$, како би се добили вектори међурезултата $\boldsymbol{\lambda}_j$, где је $1 \le j \le m_\mathrm{b}$. Након тога се рачунају контролни бити коришћењем следећег система једначина:

$$\begin{aligned} &\boldsymbol{\lambda}_1 + \mathbf{p}_1^{\mathrm{T}(S_1)} + \mathbf{p}_2^{\mathrm{T}} = 0, \\ &\boldsymbol{\lambda}_j + \mathbf{p}_j^{\mathrm{T}} + \mathbf{p}_{j+1}^{\mathrm{T}} = 0, \text{ за } j \in \{2,3,\ldots,r-1,r+1,r+2,...,m_\mathrm{b}-1\}, \\ &\boldsymbol{\lambda}_r + \mathbf{p}_1^{\mathrm{T}(S_2)} + \mathbf{p}_r^{\mathrm{T}} + \mathbf{p}_{r+1}^{\mathrm{T}} = 0, \\ &\boldsymbol{\lambda}_{m_\mathrm{b}} + \mathbf{p}_1^{\mathrm{T}(S_1)} + \mathbf{p}_{m_\mathrm{b}}^{\mathrm{T}} = 0. \end{aligned} \tag{86}$$

Сабирањем свих једначина добија се прва група контролних бита кружно померена за S_2:

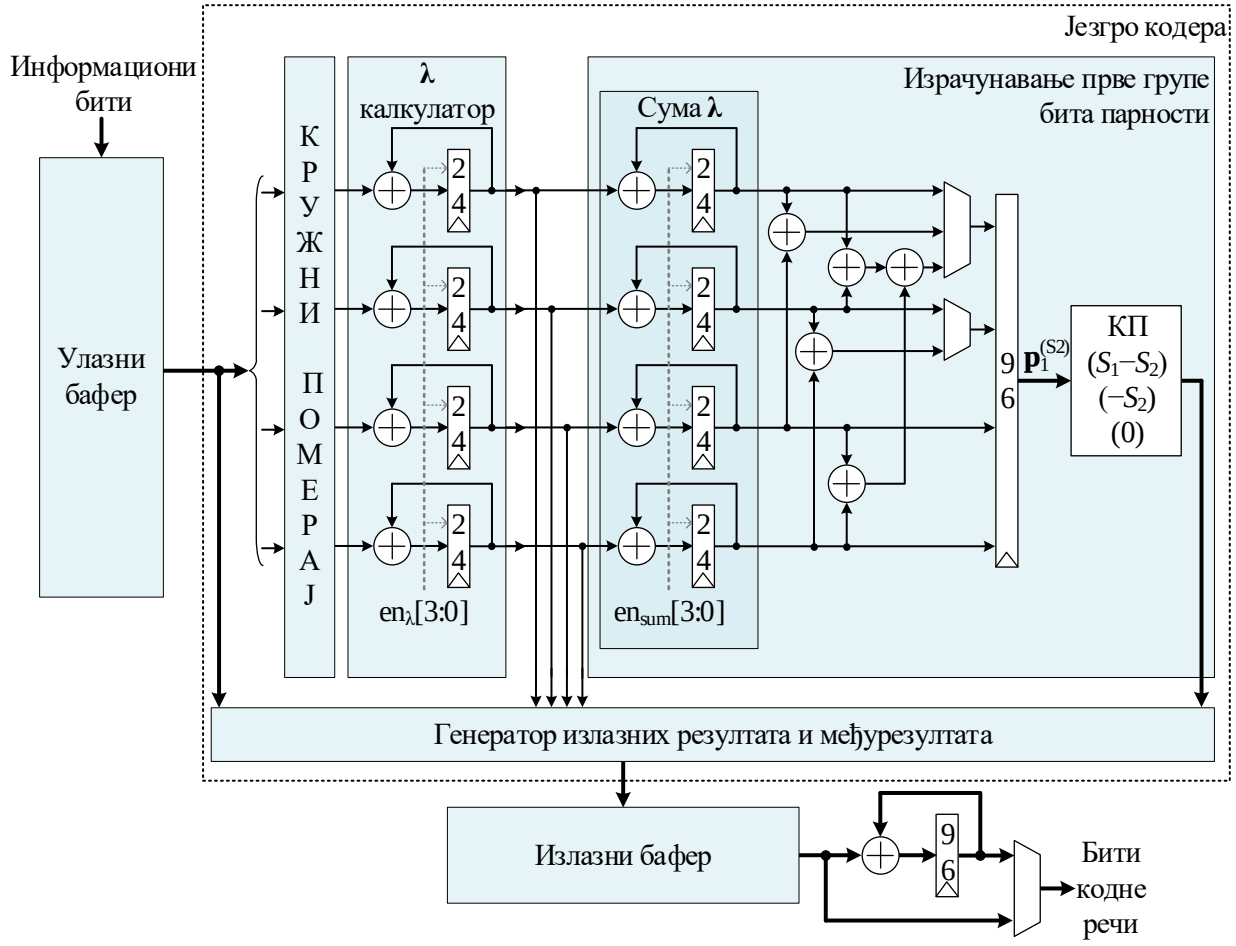
$$\mathbf{p}_1^{\mathrm{T}(S_2)} = \sum_{j=1}^{m_\mathrm{b}} \boldsymbol{\lambda}_j, \tag{87}$$

а сви остали контролни бити се могу добити рекурзивно из следећих израза:

$$\begin{aligned} &\mathbf{p}_2^{\mathrm{T}} = \mathbf{p}_1^{\mathrm{T}(S_1)} + \boldsymbol{\lambda}_1, \\ &\mathbf{p}_{j+1}^{\mathrm{T}} = \mathbf{p}_j^{\mathrm{T}} + \boldsymbol{\lambda}_j, \text{ за } j \in \{2,3,\ldots,r-2,r,r+1,...,m_\mathrm{b}-1\}, \\ &\mathbf{p}_r^{\mathrm{T}} = \mathbf{p}_1^{\mathrm{T}(S_2)} + \mathbf{p}_{r+1}^{\mathrm{T}} + \boldsymbol{\lambda}_r. \end{aligned} \tag{88}$$

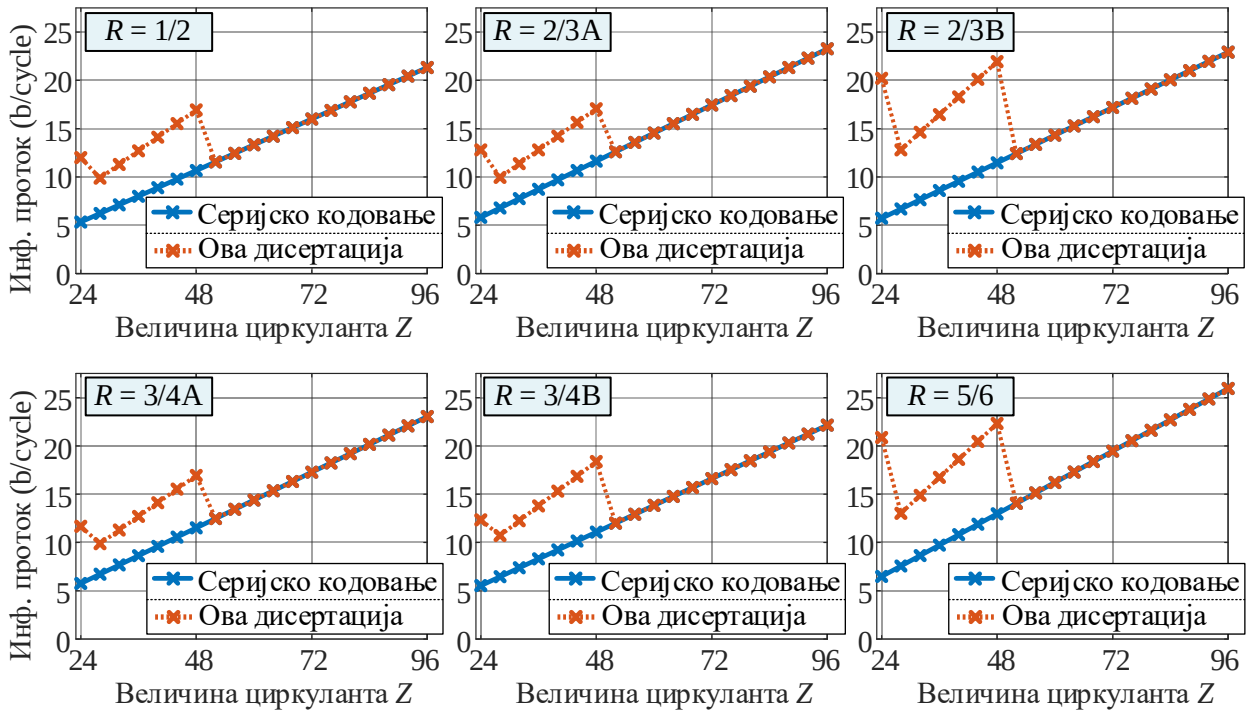
Предлог архитектуре кодера којом се имплементира описани метод кодовања приказан је на слици 3.18. Улазни део система је исти као и код 5G кодера. Информациони бити се ротирају у мрежи за кружни померај и акумулирају како би се добили вектори $\boldsymbol{\lambda}_j$. У току тог израчунавања информациони бити се прослеђују у излазни бафер. С обзиром на то да број вектора међурезултата није фиксан и може бити релативно велики, није ефикасно чувати их интерно у регистрима као код 5G кодера. Због тога је предложено да се они чувају у излазном баферу, а да се у регистрима чува само њихова сума, тј. прва група контролних бита. За израчунавање суме је задужен додатни блок за акумулацију. Међутим, ако је величина циркуланата мања од или једнака 48, више $\boldsymbol{\lambda}$ вектора се израчунава у паралели, па се њихова сума израчунава у два или четири одвојена дела. Због тога је потребно обезбедити сабирање и ових паралелно генерисаних резултата. Сумирања која следе након другог акумулационог блока на слици 3.18 служе управо томе, а мултиплексерима се селектују они резултати који одговарају величини циркуланта примењеног кода. Финални резултат тог блока је прва група контролних бита кружно померена за вредност S_2. Као што се види из израза датим у (88), тако померени бити су неопходни за израчунавање r-те групе контролних бита, па се и они морају сачувати у излазном баферу. Потребно је сачувати и прву групу контролних бита у исправном редоследу, па се за ту намену користи ротатор. На

слици је одговарајући померај обележен са $(-S_2)$ што је еквивалентно померају за $Z-S_2$ у другу страну. Поред тога, за израчунавање друге контролне групе потребан је вектор $\mathbf{p}_1^{T(S_1)}$, а он се из већ израчунатог $\mathbf{p}_1^{T(S_2)}$ може добити ротирањем за S_1-S_2. За све ове операције потребно је најмање три циклуса такта. Након тога, може се прећи на израчунавање међурезултата нове кодне речи, а контролни бити текуће кодне речи биће израчунати на излазу, из сачуваних вектора $\boldsymbol{\lambda}_j$. На излазу се из бафера читају информациони бити и прва група контролних бита, а затим се акумулацијом рачунају остале групе контролних бита. На овај начин је уведена проточна обрада у сам процес кодовања, па је проток кодера одређен трајањем израчунавања међурезултата и генерисањем прве групе контролних бита. Ипак, на кашњење утиче и време израчунавања преосталих контролних бита.



Слика 3.18. Предлог архитектуре флексибилног кодера за WiMAX стандард

Оптимизацијом редоследа процесирања циркуланата из левог дела контролне матрице (подматрица $\mathbf{H}_1$ у (85)) може се повећати проток кодера за величине циркуланата у опсегу $24 \le Z \le 48$, на исти начин како је то описано и за кодер 5G кодова. Међутим, с обзиром на то да су контролне матрице WiMAX кодова значајно мање од матрица из 5G стандарда, оптимизација генетичким алгоритмом може се заменити и случајним претраживањем редоследа процесирања. Добијени резултати се не разликују. На слици 3.19 приказани су нормализовани информациони протоци (у битима по циклусу такта) добијени серијским процесирањем и коришћењем предложене архитектуре са оптималним редоследом процесирања. Слично као и код 5G кодера, предложена архитектура даје значајно веће протоке за кодове мале и средње дужине.



Слика 3.19. Нормализовани информациони проток WiMAX LDPC кодера за различите кодне количнике и величине циркуланата

Ова анализа је показала да се малим изменама архитектуре предложене у дисертацији могу постићи слични резултати и за друге квазицикличне LDPC кодове. Додатно, предложена оптимизација редоследа процесирања даје значајна убрзања хардверских реализација кодера. Поред наведених доприноса, значајан је и метод пројектовања мреже за кружни померај који се може применити и у другим хардверским реализацијама.

4. ФЛЕКСИБИЛНИ ДЕКОДЕР LDPC КОДОВА

У овом поглављу описан је други део доприноса дисертације—реализација флексибилног декодера структурираних LDPC кодова. Суштински, предложени декодер извршава слојевито декодовање које омогућава бржу конвергенцију од симултаног декодовања. Међутим, с обзиром на то да се због проточне обраде при слојевитом декодовању најчешће јављају хазарди података, алгоритам декодовања је измењен и дато је ново решење којим се конфликти избегавају. Када конфликти постоје, потребно је уводити циклусе паузе како би се сачекао упис валидних података у меморију пре него што се приступи њиховом читању. Избегавањем конфликата, циклуси паузе су непотребни и тиме се постиже знатно већи проток и мање кашњење декодера. Додатно, тада је омогућено значајно скраћење критичне путање, увођењем великог броја степени проточне обраде, чиме се у великој мери може повећати учестаност сигнала такта, а самим тим и проток.

Нови начин декодовања назван је хибридним декодовањем, јер се LLR вредности у конфликтним ситуацијама, у одређеној мери, ажурирају као код симултаног декодовања. Као последица, перформансе контроле грешака при малом броју итерација могу бити лошије него код слојевитог декодовања. Да би се губитак у перформансама избегао (или што боље надокнадио), хибридни метод је оптимизован коришћењем генетичког алгоритма. Свеукупно, постигнуто је велико убрзање, а задржане су исте перформансе контроле грешака.

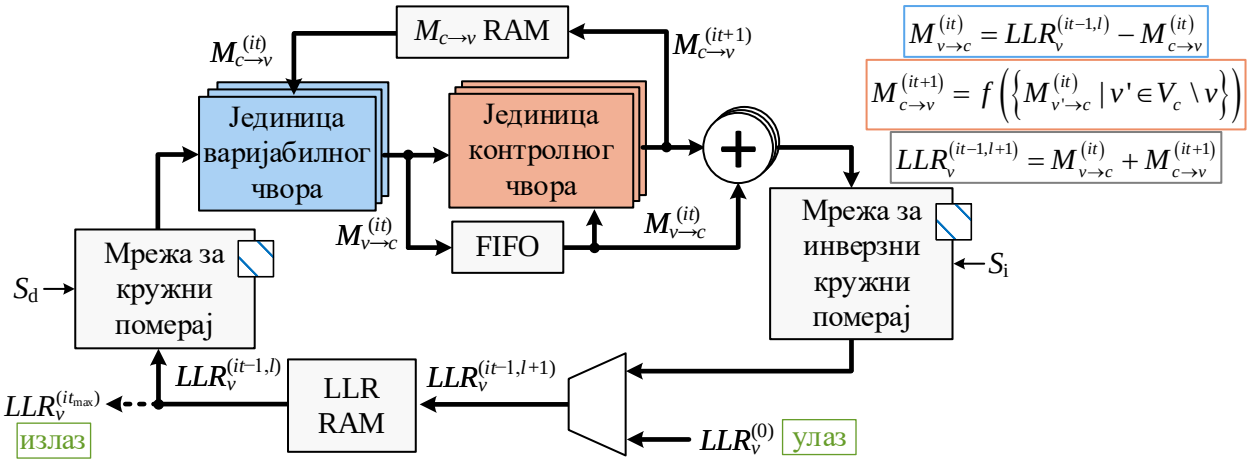
Архитектуре за хардверску реализацију декодера разликују се по степену имплементираног паралелизма. Серијске архитектуре се готово никада не имплементирају у специфичним хардверским реализацијама, јер се њима остварује мали проток. Ипак, серијски методи декодовања популарни су у софтверским реализацијама на процесорима опште намене или на дигиталним процесорима сигнала. Насупрот серијском, потпуно паралелно декодовање има потенцијал за остваривање изузетно великих протока. Међутим, његове главне мане су екстремно велико заузеће хардверских ресурса, врло често несразмерно добитку у протоку, и мала флексибилност. Стога су потпуно паралелне реализације пре свега намењене за декодере релативно мале рачунске комплексности и за кратке кодне речи. За дугачке кодове, какви су рецимо кодови из DVB фамилије стандарда или из 5G NR стандарда, разумније је користити делимично паралелне архитектуре. Најчешће, такве архитектуре обрађују један циркулант контролне матрице по циклусу такта [48], [50]–[54], [82], [129]–[134], али се оне могу додатно паралелизовати уз значајно повећање хардверских ресурса и посебне проблеме због паралелног приступа меморији [56], [135], [136].

С обзиром на то да је фокус ове дисертације хардверска реализација кодера и декодера за практичне кодове, и овде је коришћен метод у коме се процесира један циркулант контролне матрице по циклусу такта. Циљ је остварити максимално искоришћење хардверских ресурса како би се за разумну количину употребљених компоненти добио што је могуће већи проток. Стога је у овом поглављу описана и анализирана конвенционална архитектура за слојевито декодовање, како би се јасно стекао увид у проблеме на које се

наилази при њеној примени. Анализирани су временски дијаграми уписа и читања у меморије, узроци за хазарде података и неки начини за њихово решавање доступни у литератури. Након тога је описан хибридни алгоритам декодовања предложен у дисертацији и претходно објављеном раду [49], а затим и његова флексибилна хардверска реализација, као и оптимизација перформанси контроле грешака. Декодер пројектован у дисертацији користи Min-Sum алгоритам са офсетом. Након резултата и поређења са другим референтним радовима дате су додатне анализе перформанси контроле грешака, рачунске комплексности, као и нека решења која омогућавају побољшања у том погледу.

4.1. Архитектура декодера за слојевито декодовање

На слици 4.1 приказана је конвенционална архитектура којом се реализује слојевито декодовање квазицикличних кодова. Нотације на слици одговарају усвојеним нотацијама у одељку 2.4.4, а најважнији изрази су поновљени. Стања варијабилних чворова одређена су LLR вредностима, које се чувају у RAM меморији. Садржај меморије је уређен тако да се паралелно може приступити већем броју LLR-ова. Ако се обрађује један циркулант контролне матрице по такту, онда је број LLR вредности, којима се може приступити истовремено, једнак Z. Дакле, LLR вредности које одговарају целој кодној речи подељене су у n_b група по Z вредности, при чему су групе смештене у сукцесивним меморијским локацијама. Пре почетка декодовања, декодер је потребно иницијализовати почетним вредностима добијеним из демодулатора.

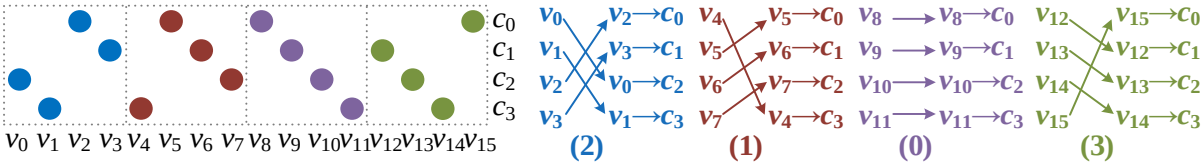


Слика 4.1. Конвенционална архитектура декодера за слојевито декодовање

Све поруке варијабилних чворова рачунају се у Z паралелних јединица (процесора) варијабилних чворова, на основу прочитаних LLR вредности и вредности порука контролних чворова у текућој итерацији. На почетку су све поруке контролних чворова једнаке нули. С обзиром на то да се у једном такту из LLR меморије читају подаци који одговарају само једној групи варијабилних чворова, онда и свака јединица варијабилног чвора генерише по једну поруку по циклусу такта и прослеђује је јединици контролног чвора. Због тога се у контролним чворовима поруке израчунавају серијски. Међутим, није могуће генерисати поруке контролног чвора све док јединица варијабилног чвора не пошаље поруке које одговарају свим варијабилним чворовима повезаним на текући контролни чвор, тј. док се не израчунају све поруке из тренутног слоја. Јединице контролних чворова рачунају поруке намењене варијабилним чворовима по истом редоследу по коме су јединице варијабилних чворова генерисале своје поруке. За израчунавање LLR вредности потребне су порука варијабилног и порука контролног чвора, због чега се поруке варијабилних чворова чувају у

FIFO баферу (енгл. *First-In-First-Out*). Њихови знакови користе се и за одређивање знака порука контролних чворова, па је потребно проследити их јединицама контролних чворова, да се не би беспотребно интерно чували.

Како би поруке варијабилних чворова у јединице контролних чворова стизале у одговарајућем распореду, потребно је извршити кружни померај LLR вредности прочитаних из меморије. Илустрација потребних ротација је приказана на слици 4.2. Сваки процесор контролног чвора одговара једној врсти контролне матрице. Слојевито декодовање се врши по врстама, па је потребно пермутовати распоред варијабилних чворова тако да се остваре одговарајуће везе у Танеровом графу. Када се заврши израчунавање нових LLR вредности, оне остају у измењеном распореду, па је неопходно урадити инверзни кружни померај пре уписа у меморију. Уписом нових LLR вредности за све варијабилне чворове у LLR RAM завршава се једна подитерација слојевитог декодовања.



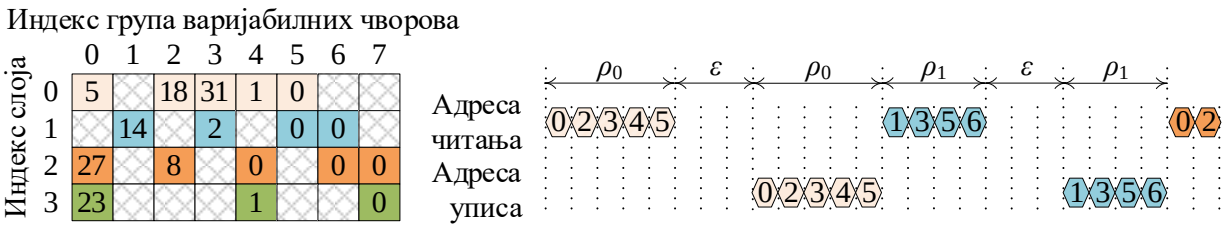
Слика 4.2. Пример једног слоја контролне матрице и одговарајуће потребне ротације у архитектури декодера за слојевито декодовање

На путањи података стоји велики број комбинационих логичких елемената. У циљу скраћења критичне путање и повећања учестаности сигнала такта, неопходно је уметнути регистре између неких логичких целина, тј. увести проточну обраду. Због тога се јавља додатно кашњење, гледано у периодама сигнала такта, од почетка процесирања слоја до његовог завршетка. Ово је илустровано временским дијаграмима адреса читања из LLR меморије и адреса уписа у LLR меморију приказаним на слици 4.3. Временски дијаграми одговарају алгоритму слојевитог декодовања, приказаном раније на слици 2.18, код кога се израчунају све поруке варијабилних чворова за један слој контролне матрице, а затим сви одговори контролних чворова из тог слоја и нове LLR вредности, па се тек онда прелази на израчунавања из наредног слоја. Пример са слике 4.3 дат је за малу пермутациону матрицу која репрезентује контролну матрицу од четири слоја. Индекс групе варијабилних чворова одговара адреси у LLR меморији.

Информациони проток претходно описаног метода може се апроксимирати са

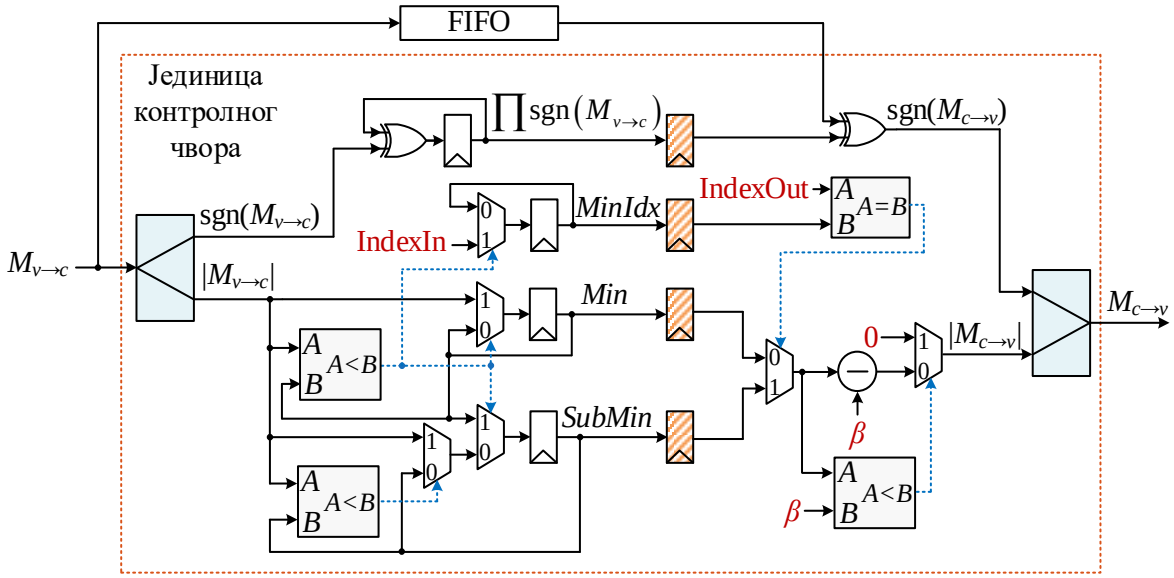
$$T=\frac{kf_{\mathrm{CLK}}}{\frac{m}{Z}\left(2\bar{\rho}+\varepsilon\right)it_{\max}+N_{\mathrm{init}}}, \tag{89}$$

где је са $\bar{\rho}$ обележена средња вредност тежина врста, са ε кашњење унето због регистара проточне обраде, а број слојева је изражен као m/Z. Параметар N_{init} представља број циклуса такта који је потребан за иницијализацију.



Слика 4.3. Временски дијаграми адреса читања из LLR RAM-а и уписа у LLR RAM код слојевитог декодовања код кога се сваки слој процесира у различитим временским интервалима

Уобичајено је да се јединица контролног чвора састоји од два, регистрима одвојена, дела, у којима се врше различита израчунавања. Први део је задужен за прикупљање порука варијабилних чворова и иницијално процесирање над њима. Пример је израчунавање минимума у Min-Sum алгоритму. Након што све поруке варијабилних чворова стигну до контролног чвора, одређени међурезултати се могу сачувати у регистрима, а онда се на основу њих секвенцијално генеришу поруке контролног чвора. С обзиром на то да се у дисертацији користи OMS алгоритам, овде је представљена његова јединица контролног чвора. На слици 4.4 приказана је њена архитектура. Поред ње, на слици је приказан и FIFO бафер за поруке варијабилних чворова, чији је излаз потребан за одређивање знака порука контролног чвора. На улазу се порука, представљена у комплементу основе два, преводи у представу у коду знак и апсолутна вредност. Знак, бинарна вредност „0” или „1”, користи се за рачунање производа свих знакова, а апсолутна вредност улази у комбинациону мрежу за одређивање тренутног минимума (*Min*). Поред минималне апсолутне вредности свих порука варијабилних чворова, потребно је и израчунати другу минималну вредност (субминимум, *SubMin*). То је неопходно урадити јер контролни чвор генерише поруку намењену једном варијабилном чвору на основу порука свих осталих варијабилних чворова. Субминимум свих вредности је заправо минимум у ситуацији када се шаље порука оном чвору чија је порука била најмања по апсолутној вредности. Како би контролни чвор знао који је то варијабилни чвор, потребно је сачувати његов идентификатор, тј. индекс (*MinIdx*). Тиме су израчунати међурезултати на основу којих се генеришу све поруке контролног чвора. У другом делу процесора контролног чвора бира се минимум или субминимум у зависности од тога да ли се порука шаље оном чвору чија је порука била најмања по апсолутној вредности или не. Затим се од резултата те селекције одузима офсет, при чему је вредност разлике ограничена са доње стране, тј. увек мора бити већа од нуле или једнака нули. Коначно, знак поруке се одређује „ексили” операцијом знака поруке варијабилног чвора коме се порука шаље и сачуваног производа свих знакова. Ово је еквивалентно израчунавању производа знакова свих порука осим једне.



Слика 4.4. Серијска јединица контролног чвора за Min-Sum алгоритам са офсетом

Уметањем додатних регистара између горенаведене две логичке целине (шрафирани регистри на слици 4.4) омогућава се паралелни рад улазног и излазног дела процесора контролног чвора. То значи да се истовремено рачунају поруке контролних чворова из једног слоја и минимуми порука варијабилних чворова из наредног слоја. Стога би читање података из LLR меморије (на основу којих се израчунавају поруке варијабилних чворова) за наредни

слој декодовања могло започети и раније, тј. пре него што се заврши упис свих LLR вредности из текућег слоја. Тиме се декодовање убрзава, под условом да се читање врши са адреса на којима се налазе ажурни подаци. Ако неке LLR вредности потребне за процесирање наредног слоја треба да буду ажуриране у току процесирања текућег слоја, неопходно је сачекати да се прво изврши упис, па тек онда наставити са читањем, како би се заобишли хазарди података услед превременог читања. На слици 4.5 приказан је псеудокод оваквог приступа. Поред њега, поновљен је и псеудокод слојевитог декодовања приказан раније на слици 2.18, ради лакшег поређења. С обзиром на то да се сада читање из LLR меморије и упис у LLR меморију извршавају паралелно, у псеудокоду са слике 4.5 ови процеси су одвојени у две нити (енгл. *threads*). Прва нит извршава читање LLR вредности и генерисање нових порука варијабилних чворова, док друга врши израчунавање порука контролних чворова и нових LLR вредности, као и њихов упис у меморију. Како би знале да

Алгоритам 2: Слојевито декодовање

Улази: контролна матрица **H**,
иницијалне вредности $\mathbf{LLR}^{(0)}$

Иницијализација:
- **for** $v = 1 : n$
  - $LLR_v = LLR_v^{(0)}$
  - **for** $c \in C_v$
    - $M_{c \to v} = 0$
- $it = 0$

Декодовање:
- **while** $it < it_{max}$ & $\mathbf{s} \neq \mathbf{0}$
  - **for** $l = 1 : (m/Z)$
    - **for** $c = l \cdot Z+1 : (l+1)Z$
      - **for** $v \in V_c$
        - $M_{v \to c} = LLR_v - M_{c \to v}$
      - **for** $v \in V_c$
        - $M_{c \to v} = f\left(\left\{M_{v' \to c} \mid v' \in V_c \setminus v\right\}\right)$
        - $LLR_v = M_{v \to c} + M_{c \to v}$
  - **for** $v = 1 : n$
    - $x_v = \dfrac{1 - \mathrm{sgn}\left(LLR_v\right)}{2}$
  - $\mathbf{s} = \mathbf{x} \cdot \mathbf{H}^{\mathrm{T}}$
  - $it = it + 1$

Излаз: вектор декодованих бита **x**

Алгоритам 3: Паралелно слојевито декодовање

Улази: контролна матрица **H**,
иницијалне вредности $\mathbf{LLR}^{(0)}$

Иницијализација:
- **for** $v = 1 : n$
  - $LLR_v = LLR_v^{(0)}$; *vUpdated*$(v) = 0$
  - **for** $c \in C_v$
    - $M_{c \to v} = 0$; *cReady*$(c) = 0$
- $it_1 = 0$; $it_2 = 0$

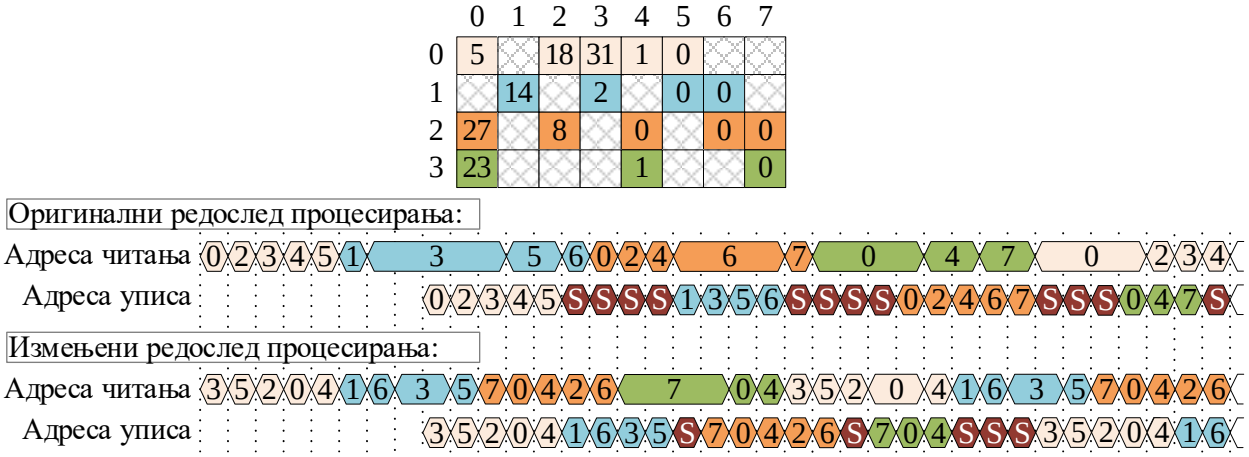
Декодовање:
- **ReadThread: while** $it_1 < it_{max}$ & $\mathbf{s} \neq \mathbf{0}$
  - **for** $l = 1 : (m/Z)$
    - **for** $c = l \cdot Z+1 : (l+1)Z$
      - **for** $v \in V_c$
        - **wait until** *vUpdated*$(v) =$ ***true***
        - $M_{v \to c} = LLR_v - M_{c \to v}$
        - *vUpdated*$(v) =$ ***false***
      - *cReady*$(c) =$ ***true***
  - $it_1 = it_1 + 1$
- **WriteThread: while** $it_2 < it_{max}$ & $\mathbf{s} \neq \mathbf{0}$
  - **for** $l = 1 : (m/Z)$
    - **for** $c = l \cdot Z+1 : (l+1)Z$
      - **wait until** *cReady*$(c) =$ ***true***
      - **for** $v \in V_c$
        - $M_{c \to v} = f\left(\left\{M_{v' \to c} \mid v' \in V_c \setminus v\right\}\right)$
        - $LLR_v = M_{v \to c} + M_{c \to v}$
        - *vUpdated*$(v) =$ ***true***
      - *cReady*$(c) =$ ***false***
    - **for** $v = 1 : n$
      - $x_v = \left(1 - \mathrm{sgn}\left(LLR_v\right)\right) / 2$
    - $\mathbf{s} = \mathbf{x} \cdot \mathbf{H}^{\mathrm{T}}$
    - $it_2 = it_2 + 1$

Излаз: вектор декодованих бита **x**

Слика 4.5. Псеудокод алгоритма слојевитог декодовања без и са истовременим читањима и уписима LLR вредности

ли је потребно сачекати ажурирање LLR вредности или генерисање међурезултата у контролним чворовима, нити комуницирају преко помоћних вектора *vUpdated* и *cReady*.

Временски дијаграми на истом примеру пермутационе матрице као са слике 4.3, али за истовремено читање и упис LLR вредности, приказани су на слици 4.6. Очекивано, декодовање је брже, али се може уочити велики број циклуса паузе (енгл. *stall cycles*) обележених словом „S”. Међутим, процесирање контролне матрице се не мора радити по оригиналном редоследу. Дозвољено је изменити редослед процесирања циркуланата унутар врсте тако да се број конфликтних ситуација смањи и избегне додавање циклуса паузе, па је и један пример измењеног редоследа процесирања приказан на слици 4.6. Такође, може се и мењати редослед процесирања слојева, чиме се даје још један степен слободе за оптимизацију редоследа процесирања и постизање бржег рада декодера [48], [50]–[52]. Ипак, поготово ако је дубина проточне обраде велика (на слици 4.6 број степени проточне обраде је свега три), у општем случају није могуће у потпуности укинути циклусе паузе [49].



Слика 4.6. Временски дијаграми адреса читања из LLR RAM-а и уписа у LLR RAM код слојевитог декодовања при оригиналном и редоследу процесирања измењеном за краће време декодовања

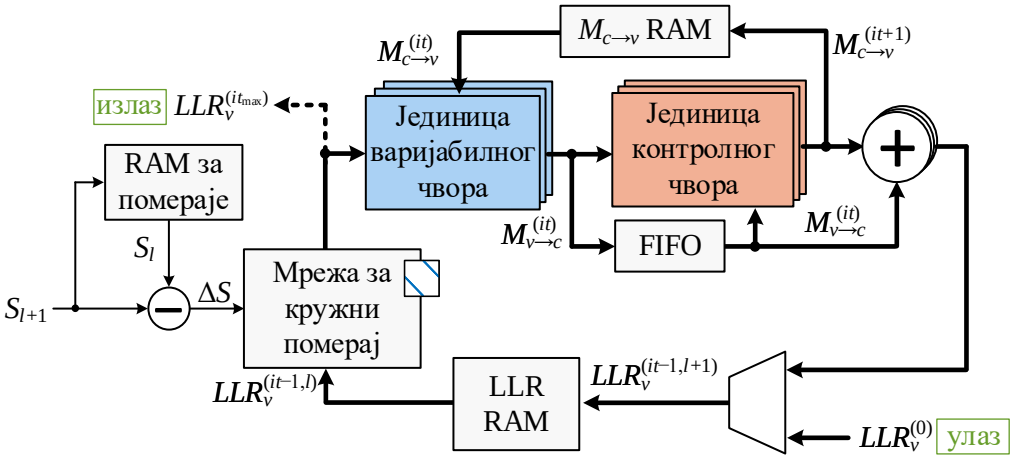
Информациони проток на овај начин паралелизованог декодера за слојевито декодовање се сада може изразити са

$$T = \frac{kf_{\mathrm{CLK}}}{\frac{m}{Z}\cdot\bar{\rho}\cdot it_{\max} + N_{\mathrm{stall}} + N_{\mathrm{init}}} = \frac{kf_{\mathrm{CLK}}}{N_{\mathrm{circ}}\cdot it_{\max} + N_{\mathrm{stall}} + N_{\mathrm{init}}}, \tag{90}$$

где је N_{stall} укупан број циклуса паузе уведених због избегавања конфликата, а N_{circ} укупан број циркуланата у контролној матрици.

Поред измене редоследа процесирања, смањење броја циклуса паузе може се постићи и изменом архитектуре декодера. Број циклуса паузе директно зависи од дубине проточне обраде, па се смањењем броја степени проточне обраде смањује и број циклуса паузе. Међутим, за исту критичну путању, мањи број степени проточне обраде имплицира и нижу максималну учестаност такта. Ипак, укидањем неких логичких елемената на критичној путањи, за исту учестаност такта могуће је поставити мањи број степени проточне обраде. Један такав приступ приказан је на слици 4.7. У приказаној архитектури декодера користи се само једна мрежа за кружни померај. На самом почетку, вредности помераја су исте као и код мреже за директни кружни померај декодера са слике 4.1. Међутим, пошто се LLR вредности већ након првог слоја прве итерације уписују у меморију у измењеном редоследу, сваки наредни пут када је потребан њихов кружни померај, оне се померају за разлику жељеног и већ оствареног помераја. Вредности тренутних помераја се чувају у посебном RAM-у. Када се заврши декодовање, пре прослеђивања финалних LLR вредности, потребно

је вратити их у исправан редослед, што се може урадити већ доступном мрежом за кружни померај у току уписа LLR вредности за наредну кодну реч. Алгоритамске технике за смањење броја или потпуно укидање циклуса паузе описане су у потпоглављу 4.2.



Слика 4.7. Архитектура декодера за слојевито декодовање која користи само једну мрежу за кружни померај [81]

На проток декодера, дат изразима (89) и (90), утиче и време иницијализације. За кратке кодне речи и кодове код којих је величина циркуланата мала, то време обично није велико и најчешће је једнако n_b. Другим речима, могуће је уписати целу групу од Z LLR вредности у једном циклусу такта. Ширина података на улазном интерфејсу би у том случају била $b_{\text{in}} = b_{\text{LLR}} Z$, где је b_{LLR} битска ширина LLR вредности. Нека је $b_{\text{LLR}} = 8$. Тада би, рецимо за $Z = 32$, ширина податка на улазном интерфејсу морала да буде 256 бита. Дакле, већ при малим вредностима величине циркуланта, ширина улазног интерфејса је велика. За највећу вредност циркуланта у 5G NR стандарду, битска ширина би била 3072 бита, што ни један практичан екстерни систем не може да подржи. Ако би се задржала мања ширина података на интерфејсима, време иницијализације (у циклусима такта) би било

$$N_{\text{init}} = n_b \frac{b_{\text{LLR}} Z}{b_{\text{in}}}, \tag{91}$$

што може да буде изузетно велики број. Као пример, за најдужу кодну реч из 5G NR стандарда (основни граф 1, $R = 22/66$) и вредности $b_{\text{LLR}} = 8$ и $b_{\text{in}} = 256$, број циклуса за иницијализацију би био 816, а с обзиром на то да је број циркуланата одговарајућег кода једнак 316, може се закључити да иницијализација траје скоро колико и три итерације декодовања, што драстично смањује проток, посебно при малом броју итерација. За веће кодне количнике, овај проблем је још израженији (за $R = 22/55$: $N_{\text{circ}} = 79$ и $N_{\text{init}} = 324$, тј. преко четири итерације декодовања).

Због свега наведеног, потребно је омогућити двоструко баферисање улазних података, тј. обезбедити још једну меморију, у коју ће се уписивати подаци нове кодне речи, у току декодовања прве кодне речи. Организација друге меморије треба да буде потпуно иста као и организација меморије која се користи за декодовање. Захваљујући томе, декодер може, након завршетка декодовања прве кодне речи, да преузме контролу над другом меморијом, у којој су већ иницијализоване LLR вредности друге кодне речи, и да надаље њу користи за декодовање. Тада прва меморија постаје бафер из кога се преко једног порта читају декодовани подаци, а преко другог порта уписује наредна, трећа, кодна реч. Докле год је проток интерфејса већи од протока језгра декодера, параметар N_{init} је једнак нули [129].

4.2. ХИБРИДНИ АЛГОРИТАМ ДЕКОДОВАЊА

Потпуно укидање броја циклуса паузе у општем случају може се постићи изменом алгоритма декодовања. Све алгоритамске измене присутне у литератури, као и алгоритам предложен у дисертацији, подразумевају да се у конфликтним ситуацијама не чека на ажурирање LLR вредности, већ да се читају вредности које још увек нису ажуриране (у даљем тексту зване неажурним LLR вредностима). Међутим, како се перформансе контроле грешака не би екстремно нарушиле, доприноси контролних чворова додају се на другачији начин него код стандардног слојевитог декодовања.

Ажурирање LLR вредности код слојевитог декодовања дато је изразом (64), мада се комбиновањем са изразом (63), може записати и у другачијој форми:

$$LLR_v^{(it-1,l+1)} = M_{v\to c}^{(it)} + M_{c\to v}^{(it+1)} = \left(LLR_v^{(it-1,l)} - M_{c\to v}^{(it)}\right) + M_{c\to v}^{(it+1)} = LLR_v^{(it-1,l)} + \Delta M_{c\to v}, \tag{92}$$

где је са $\Delta M_{c\to v}$ означена разлика између порука контролних чворова у текућој и у претходној итерацији. Суштински, разлика $\Delta M_{c\to v}$ представља допринос контролног чвора c LLR вредности варијабилног чвора v. Због тога је неопходно уписати нову LLR вредност пре њеног читања у неком од наредних слојева декодовања. У супротном, допринос контролног чвора остаје изгубљен чиме перформансе контроле грешака постају знатно лошије.

Вредност $LLR_v^{(it-1,l)}$ једнака је $LLR_v^{(it-1,l-k)}$ ако између слоја са индексом $l-k$ и слоја са индексом l ни у једном слоју није постојао циркулант кога пресеца колона v. У даљем излагању, идентификатори $(it-1,l)$ и $(it-1,l+1)$, коришћени у изразима (64) и (92), који означавају вредности након слоја са индексима l и $l+1$ у текућој итерацији, замењени су општим ознакама l_i и l_j, као у следећем изразу:

$$LLR_v^{(l_j)} = LLR_v^{(l_i)} + \Delta M_{c\to v}. \tag{93}$$

Сада $LLR_v^{(l)}$ означава LLR вредност која је последњи пут ажурирана у слоју са индексом l, где l може да узима вредности од 1 до $it_{\max}\cdot m/Z = it_{\max}\cdot m_{\mathrm{b}}$.

Ако два контролна чвора, c' из слоја l_j и c'' из слоја $l_k > l_j$, доприносе LLR вредности за варијабилни чвор v истовремено, ажурирање се може изразити са

$$LLR_v^{(l_k)} = LLR_v^{(l_i)} + \Delta M_{c'\to v} + \Delta M_{c''\to v}, \tag{94}$$

где је $LLR_v^{(l_i)}$ последња ажурирана вредност након слоја $l_i < l_j$. Овај начин ажурирања LLR вредности се може искористити за укидање хазарда података услед превременог читања. Наиме, ако настане конфликт због кога би се код слојевитог декодовања увели циклуси паузе, није неопходно чекати на ажурирање LLR вредности. Могуће је прочитати стару вредност ($LLR_v^{(l_i)}$) и њу користити за израчунавање поруке варијабилног чвора, али касније ажурирање LLR вредности реализовати као у једначини (94). Дакле, у том случају су поруке варијабилног чвора

$$M_{v\to c'}^{(it)} = LLR_v^{(l_i)} - M_{c'\to v}^{(it)}, \tag{95}$$

што је иста вредност као код слојевитог декодовања, и

$$M_{v\to c''}^{(it)} = LLR_v^{(l_i)} - M_{c''\to v}^{(it)}, \tag{96}$$

што је вредност која се разликује јер би, у случају да су направљени циклуси паузе, ова порука имала вредност

$$M_{v\to c"}^{(it)} = LLR_v^{(l_j)} - M_{c"\to v}^{(it)} = LLR_v^{(l_i)} + \Delta M_{c'\to v} - M_{c"\to v}^{(it)}. \tag{97}$$

С обзиром на то да варијабилни чвор истовремено генерише више од једне поруке које шаље различитим контролним чворовима, описано понашање је делимично карактеристично за симултано декодовање.

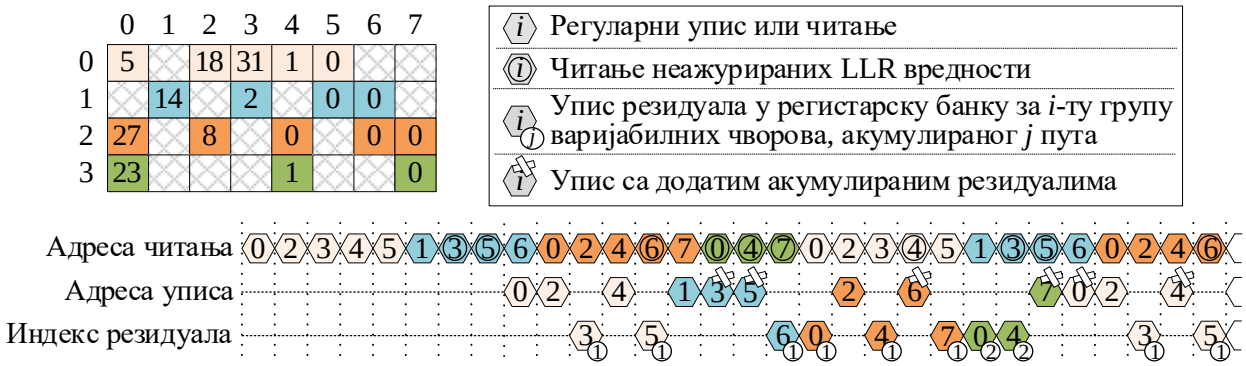
У техничком извештају [55] предложено је да се ажурирање варијабилних чворова увек ради као у (93). Није предложена архитектура декодера којом је то могуће остварити, али се из описа алгоритма може закључити да је основна мана таквог метода неопходност чувања копија LLR вредности у одвојеној меморији. Наиме, за генерисање порука варијабилних чворова потребно је читати LLR вредности, а према (93) неопходно је и са других меморијских локација читати вредности потребне за ажурирање, што је једино могуће ако постоји одвојена меморија у којој се налазе копије.

У раду [56] предложено је да се, у случајевима када постоје конфликти, прочитају старе LLR вредности и на основу њих генеришу поруке варијабилних чворова, али се доприноси контролних чворова из претходних слојева чувају у одвојеној регистарској банци и накнадно додају на LLR вредности добијене у текућем слоју:

$$LLR_v^{(l_k)} = \left(M_{v\to c"}^{(it)} + M_{c"\to v}^{(it+1)}\right) + \Delta M_{c'\to v}. \tag{98}$$

Према томе, упис LLR вредности се одлаже, све док се не ослободи временски тренутак у коме је могуће ажурирати их, тако да за се сви акумулирани доприноси не изгубе при наредном ажурирању. То значи да се упис нових LLR вредности врши само ако до тренутка уписа, приликом процесирања наредних слојева, поново нису прочитане старе LLR вредности, за исту групу варијабилних чворова. На овај начин су циклуси паузе у потпуности елиминисани. Међутим, ако је број степени проточне обраде велики, конфликтне ситуације се јављају изузетно често, па се одлагање уписа за исту групу варијабилних чворова може десити више од једанпут. Екстреман случај је да се упис никада и не изврши. Због тога су перформансе контроле грешака значајно нарушене, па се и овде мора водити рачуна о броју степени проточне обраде, јер ће мала дубина проточне обраде довести до мањег одлагања и бољих перформанси.

Временски дијаграми који описују рад описаног декодера приказани су на слици 4.8. Аутори рада [56] доприносе контролних чворова називају резидуалима (енгл. *residue*), па је на слици усвојена та номенклатура. Поред адреса читања и уписа, на дијаграмима је приказан и индекс резидуала, тј. индекс групе варијабилних чворова за коју је неопходно сачувати доприносе контролних чворова у регистарској банци.



Слика 4.8. Временски дијаграми декодовања заснованог на акумулацији доприноса контролних чворова (резидуала) [56] – дијаграми преузети из [49]

У дисертацији је предложен метод за у основи слојевито декодовање код кога се уписи LLR вредности не одлажу, а притом се задржавају сви доприноси контролних чворова. На овај начин се омогућава увођење произвољног броја степени проточне обраде, при чему се декодовање и даље може урадити без превеликог губитка у перформансама контроле грешака. Као и код два претходно описана метода, и у овом методу се у случају конфликта не чека на ажурирање LLR вредности, већ се најсвежије вредности, које су доступне у меморији, користе за израчунавање порука варијабилних чворова као у (96). Када се заврши израчунавање LLR вредности из претходних слојева, оне се уписују у LLR RAM. Додатно, уписују се и у одвојен бафер за ажурирање из текућег слоја, ако су варијабилни чворови генерисали поруке на основу неажурних вредности. Поменуто ажурирање се реализује као у (93). У свим осталим случајевима, ажурирање је као код стандардног слојевитог декодовања, тј. као у (64). На овај начин се постиже да је учесталост уписа у LLR RAM једнака као код класичног слојевитог декодовања. С обзиром на то да неки варијабилни чворови генеришу више порука ка различитим контролним чворовима на основу исте LLR вредности, декодовање није у потпуности слојевито, па је у дисертацији названо хибридним декодовањем. Псеудокод алгоритма хибридног декодовања приказан је на слици 4.9. Нит ReadThread и нит WriteThread су временски померене за онолико времена колико је потребно да се израчунају све поруке варијабилних чворова из првог слоја прве итерације.

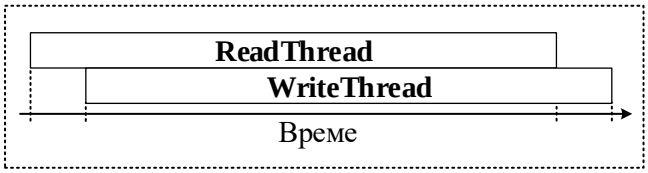
Алгоритам 4: Хибридно декодовање

Улази: контролна матрица **H**,
иницијалне вредности $\mathbf{LLR}^{(0)}$

Иницијализација:

for $v = 1 : n$
- $LLR_v = LLR_v^{(0)}$; $noReads(v) = 0$
- **for** $c \in C_v$
  - $M_{c\to v}^{(0)} = 0$; $conflict_{v\to c} = 0$

$it_1 = 0$; $it_2 = 1$



Декодовање:

ReadThread: while $it_1 < it_{max}$ & $\mathbf{s} \neq \mathbf{0}$
- **for** $l = 1 : (m/Z)$
  - **for** $c = l \cdot Z+1 : (l+1)Z$
    - **for** $v \in V_c$
      - $M_{v\to c}^{(it_1)} = LLR_v - M_{c\to v}^{(it_1)}$
      - $conflict_{v\to c} = noReads(v)$
      - $noReads(v)$ += 1
- $it_1 = it_1 + 1$

WriteThread: while $it_2 \leq it_{max}$ & $\mathbf{s} \neq \mathbf{0}$
- **for** $l = 1 : (m/Z)$
  - **for** $c = l \cdot Z+1 : (l+1)Z$
    - **for** $v \in V_c$
      - $M_{c\to v}^{(it_2)} = f\left(\left\{M_{v'\to c}^{(it_2-1)} \mid v' \in V_c \setminus v\right\}\right)$
      - **if** $conflict_{v\to c} = 0$ **then**
        - $LLR_v = M_{v\to c}^{(it_2-1)} + M_{c\to v}^{(it_2)}$
      - **else**
        - $LLR_v = LLR_v + M_{c\to v}^{(it_2)} - M_{c\to v}^{(it_2-1)}$
      - $noReads(v)$ −= 1
- **for** $v = 1 : n$
  - $x_v = \left(1 - \mathrm{sgn}\left(LLR_v\right)\right)/2$
- $\mathbf{s} = \mathbf{x} \cdot \mathbf{H}^{\mathrm{T}}$
- $it_2 = it_2 + 1$

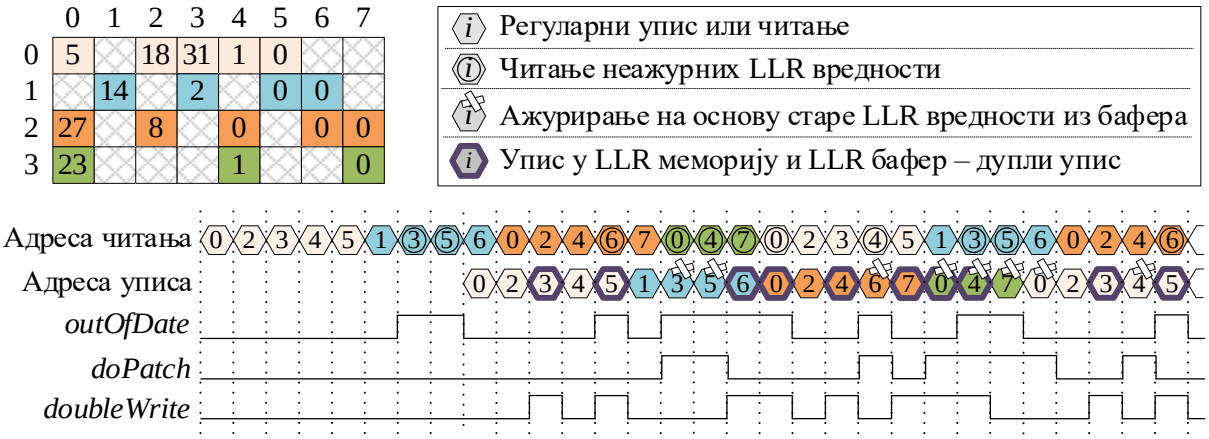
Излаз: вектор декодованих бита **x**

Слика 4.9. Псеудокод алгоритма хибридног декодовања [49]

Важно је напоменути да чување LLR вредности потребних за нека ажурирања захтева одвојен бафер за податке, што наликује претходно поменутом методу из [55]. Међутим, читање LLR вредности за ажурирање варијабилних чворова обавља се само онда када је потребно разрешити конфликт. У свим осталим ситуацијама, ажурирање се реализује као код

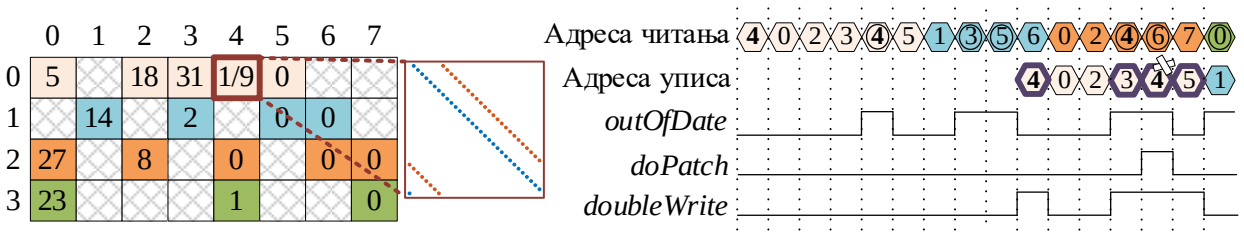
класичног слојевитог декодовања. Такав приступ омогућава да је бафер много мањи него у методу из [55]. Што је још важније, портови бафера за упис и за читање нису стално заузети. То је ефикасно искоришћено у хардверској имплементацији, где се за реализацију бафера користе већ доступни ресурси, који су тренутно неактивни. У реализацији на FPGA чипу стога нема додатних меморијских ресурса, који су последица увођења овог бафера, што је детаљно описано у потпоглављу 4.3.

На слици 4.10 приказани су временски дијаграми адреса уписа и читања LLR вредности за декодер који обавља хибридно декодовање. Посебно су означене адресе читања неажурних LLR вредности. Такође, посебно су означене и адресе уписа LLR вредности које је потребно ажурирати коришћењем (93), тј. за које је потребно прочитати старе вредности из одвојеног бафера. Да би оне биле доступне у баферу, неопходно их је уписати, па су и ти циклуси обележени на слици 4.10. Сигнале *outOfDate*, *doPatch* и *doubleWrite* генерише контролна јединица и њима се контролишу одговарајући процесирајући елементи декодера детаљно описани у потпоглављу 4.3.



Слика 4.10. Временски дијаграми хибридног декодовања [49]. Сигнал *outOfDate* означава да је прочитана неажурна LLR вредност, сигнал *doPatch* да је потребно радити ажурирање на основу старе LLR вредности из бафера, а сигнал *doubleWrite* да је податке потребно уписати и у бафер.

Хибридним декодовањем се решава и проблем двоструких циркуланата који постоје код структурираних IRA кодова из DVB-S2 стандарда. Двоструки циркулант се састоји од збира две кружно померене јединичне матрице. Стога је његово процесирање потребно раздвојити у два циклуса такта, при чему се у првом циклусу обрађује прва кружно померена јединична матрица, а у другом циклусу друга. Јасно је да су LLR вредности прочитане у другом циклусу неажурне, па се њихово ажурирање мора урадити коришћењем LLR вредности које се упишу након процесирања првог циркуланта. Због потенцијалних конфликата у приступима меморији, пожељно је, између два циклуса, процесирати друге циркуланте из слоја. Пример временског дијаграма приказан је на слици 4.11.



Слика 4.11. Пример хибридног декодовања када је у контролној матрици присутан двоструки циркулант

Информациони проток декодера који користи хибридно декодовање је

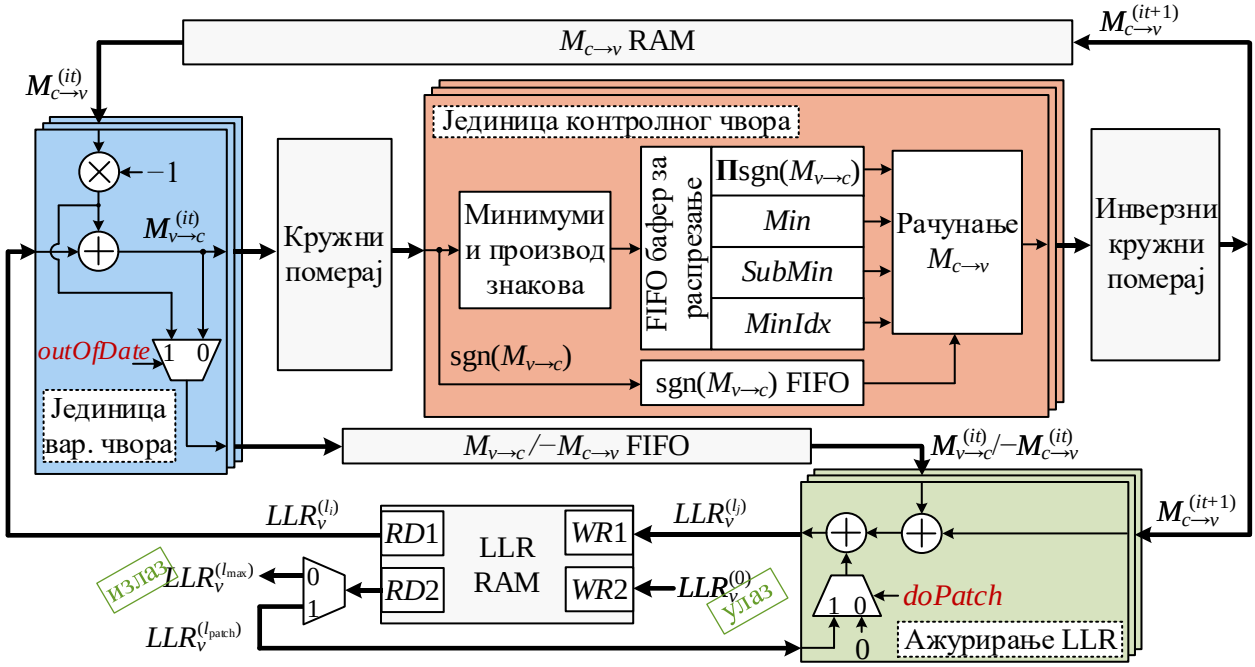
$$T = \frac{kf_{\mathrm{CLK}}}{N_{\mathrm{circ}} \cdot it_{\max} + N_{\mathrm{init}}}. \tag{99}$$

Дакле, у изразу су потпуно уклоњени циклуси паузе. Број циклуса потребних за иницијализацију се такође може свести на нулу двоструким баферовањем LLR вредности.

4.3. Хардверска реализација декодера за хибридно декодовање

4.3.1. Архитектура декодера

Детаљна архитектура декодера за хибридно декодовање приказана је на слици 4.12. RAM за LLR вредности је реализован помоћу два независна блока двопортних меморија како би се обезбедило дупло баферисање. У првом меморијском блоку налазе се LLR вредности кодне речи која се тренутно декодује, док други блок служи за упис наредне кодне речи и читање претходно декодоване кодне речи. Након завршетка декодовања, ова два блока мењају улоге, па други блок постаје меморија за декодовање, а први меморија за бафер. Оваква замена контекста омогућава да је време иницијализације из једначине (99) једнако нули. Описани пар меморијских блокова је на слици 4.12 представљен једним блоком са два посебна порта за читање (*RD*1 и *RD*2) и два порта за упис (*WR*1 и *WR*2), који се интерно повезују на одговарајућу меморију према горенаведеним правилима. Поред раније наведених функционалности, некоришћене локације у бафер меморији користе се за чување LLR вредности потребних за ажурирање услед конфликтних ситуација, о чему ће више бити речи у наставку.



Слика 4.12. Архитектура декодера за хибридно декодовање [49]

Кодна реч која се уписује у бафер меморију и кодна реч која се из ње чита могу бити у одвојеним меморијским просторима. Међутим, то није неопходно, ако се води рачуна о томе

да се забрани упис нових података на меморијске локације које још увек нису прочитане. Комуникацију са спољашњим уређајима декодер обавља преко стриминг интерфејса (енгл. *streaming interface*) са контролом тока података. Стога се може увести додатна комуникација између улазног и излазног модула којом се омогућава коришћење истог адресног простора за нову и стару кодну реч. Излаз декодера не морају бити само знаци LLR вредности (тврде одлуке), већ је предвиђено да се читају меке одлуке, дакле сви бити LLR вредности.

Након иницијализације, декодовање започиње читањем LLR вредности и њиховим прослеђивањем јединицама варијабилних чворова у којима се израчунавају поруке варијабилних чворова. У првој итерацији су све поруке контролних чворова, које се иначе чувају у одвојеној меморији ($M_{c\to v}$ RAM), постављене на нулу. Поруке варијабилних чворова прослеђују се јединицама контролних чворова, а како су потребне и за ажурирање LLR вредности, уписују се и у посебан FIFO бафер. У случају да су из меморије за декодовање прочитане LLR вредности које нису ажурне, у FIFO бафер се прослеђују поруке контролних чворова, јер се у том случају ажурирање ради помоћу њих, а не помоћу порука варијабилних чворова (видети алгоритам 4 са слике 4.9). Како би блокови за ажурирање LLR вредности били једноставнији, у FIFO бафер се смештају негативне вредности порука контролних чворова. Овакво коришћење FIFO бафера је раније предложено у [82] за решавање проблема двоструког циркуланта у DVB-S2 кодовима. Одабир вредности, које треба да буду прослеђене у FIFO бафер, врши се помоћу сигнала *outOfDate*, чије је генерисање описано на слици 4.10.

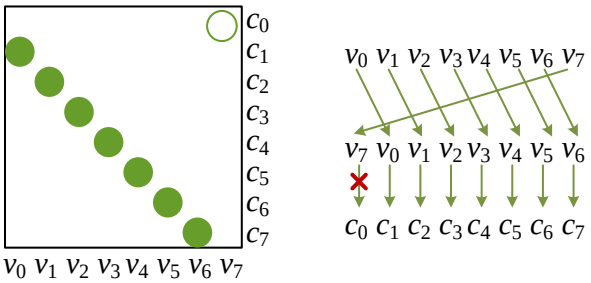
Поруке варијабилних чворова се пермутују у мрежи за кружни померај пре уласка у јединице контролних чворова. Извођење пермутације порука, уместо пермутације LLR вредности, која је коришћена у архитектури са слике 4.1, доприноси мањем заузећу хардверских ресурса, јер поруке обично захтевају мањи број бита за ефикасну представу него што је то случај са LLR вредностима [56]. Ово је веома значајно за флексибилне реализације декодера које треба да подрже велики број величина циркуланата, какве су на пример реализације за Wi-Fi, WiMAX или 5G NR стандард. У дисертацији, реализована су три различита декодера за WiMAX, 5G NR и DVB-S2(x) стандарде. Међутим, декодер може да декодује било који квазицикличан код чија је величина циркуланта подржана мрежом за кружни померај. Флексибилне мреже за кружни померај за WiMAX и 5G NR комуникационе стандарде описане су у одељку 3.3.1. У конкретној реализацији декодера, коришћене су мреже које могу да померају само један блок улазних података, тј. није предвиђено процесирање више од једног циркуланта за кратке кодове. Једина разлика у односу на мреже описане у одељку 3.3.1 је у томе што је у декодеру потребно ротирати вишебитне улазне податке. Ротатор за DVB-S2(x) стандард не мора да буде флексибилан, јер је величина циркуланата фиксирана на вредност 360. Зато је он реализован уз помоћ три фиксна комбинациона степена, који врше померај података у умношцима бројева 45, 8 и 1, респективно. На основу примљених порука варијабилних чворова, јединице контролних чворова генеришу своје поруке, које је потребно инверзно пермутовати, како би у исправном редоследу биле прослеђене у блокове за ажурирање LLR вредности, али и уписане у меморију из које се читају у наредној итерацији декодовања.

У зависности од тога да ли су из меморије за декодовање прочитане ажурне LLR вредности, њихово ажурирање након процесирања у јединицама контролних чворова може бити урађено на два начина. У блок за ажурирање LLR вредности стижу нова генерисана порука контролног чвора и вредност из FIFO бафера. С обзиром на то да она може бити порука варијабилног чвора или негативна вредност поруке контролног чвора из претходне итерације, резултат збира је или нова LLR вредност или допринос контролног чвора $\Delta M_{c\to v}$. У другом случају, потребно је прочитати старе неажурне LLR вредности ($LLR^{l_{\text{patch}}}$) из бафера и на њих додати доприносе контролних чворова. Сигнал *doPatch*, који се генерише као што је описано на слици 4.10, контролише да ли ће у коначни збир ући неажурна LLR вредност

или не. Дакле, у одређеним тренуцима декодовања, излазни порт *RD*2 преко кога се чита већ декодована претходна кодна реч, мора се искористити за читање неажурних LLR вредности. За то време могуће је да блок за комуникацију са спољашњим уређајима (није приказан на слици 4.12) заустави слање података, ако се мора сачекати да се ослободи порт *RD*2. Међутим, постоји велики број слободних циклуса такта када се подаци старе кодне речи могу прочитати, па се не успорава рад целог декодера.

Приликом уписа у меморију за декодовање проверава се да ли су са локације на коју је потребно извршити упис прочитане неажурне LLR вредности за генерисање порука из неког од наредних слојева контролне матрице. Ако је то случај, нове генерисане LLR вредности се уписују и у бафер, па је због тога ова ситуација названа двоструким уписом, а контролише се сигналом *doubleWrite*, чије је генерисање описано на слици 4.10. Слично као и код коришћења бафера за читање старих LLR вредности, при двоструком упису се за упис у бафер преузима порт преко кога се уписују LLR вредности за наредну кодну реч. У тим ситуацијама је понекад потребно зауставити пријем нових података, али као и раније, постоји велики број циклуса такта када је порт *WR*2 слободан за упис LLR вредности нове кодне речи.

У случају да је потребно процесирати непотпуни циркулант који постоји у контролној матрици свих структурираних IRA кодова (слика 2.9), неопходно је омогућити да се задржи потпуно иста архитектура, али да постојање овакве структуре не деградира перформансе декодовања. Постојање непотпуног циркуланта (чији је облик приказан на слици 4.13) значи да је веза између једног варијабилног и једног контролног чвора раскинута. Стога је потребно раскинути везу и у хардверској реализацији декодера. То је урађено тако што је једном процесору варијабилног чвора омогућено да пошаље поруку максималне вредности, без обзира на то шта је резултат претходних израчунавања. Како се у декодеру користи Min-Sum алгоритам декодовања, ова порука не утиче на израчунавања у контролном чвору, јер он израчунава минимуме апсолутних вредности свих примљених порука. Поред тога, приликом уписа ажурираних LLR вредности, потребно је прескочити вредност за варијабилни чвор који није повезан. Пошто се на истој меморијској локацији налазе вредности за остале варијабилне чворове, упис се мора извршити, али је у једној јединици за ажурирање омогућено да проследи стару вредност, претходно прочитану из меморије за бафер. Због тога је неопходно и за ову специфичну ситуацију унапред извршити двоструки упис.

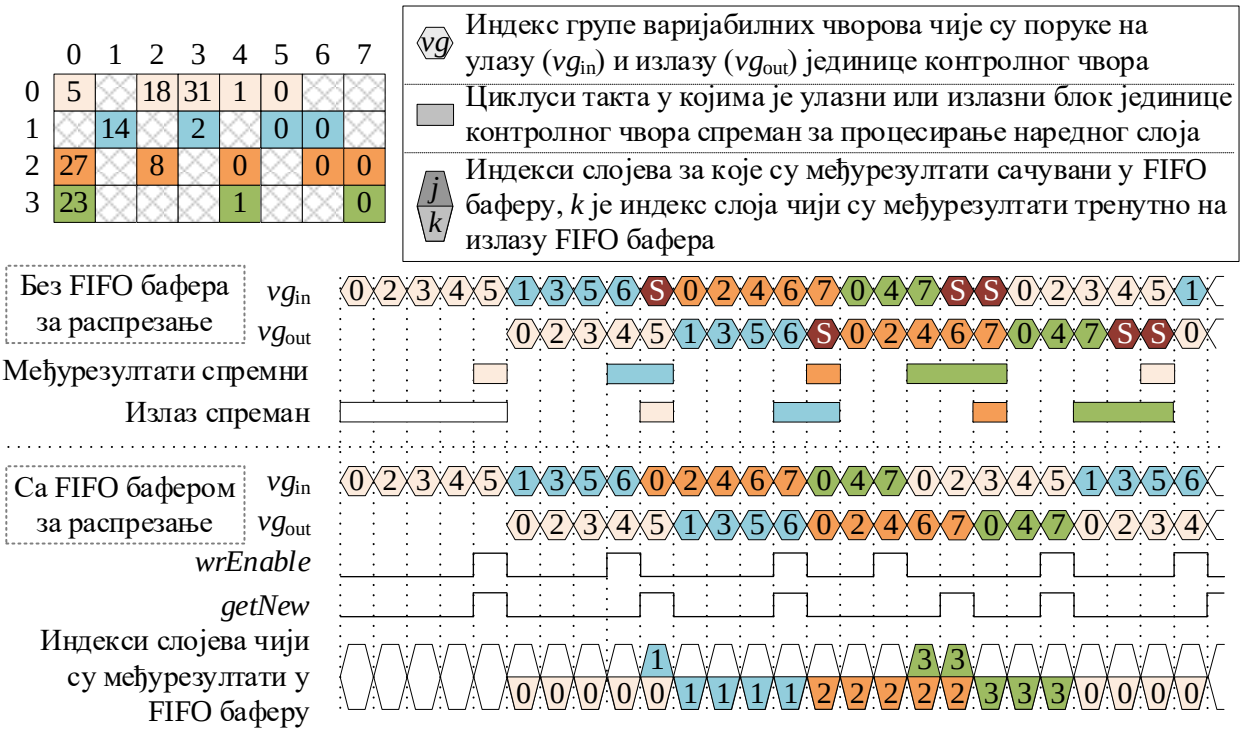


Слика 4.13. Структура непотпуног циркуланта

4.3.2. РАЗРЕШЕЊЕ ХАЗАРДА ПОДАТАКА У ПРОЦЕСОРУ КОНТРОЛНОГ ЧВОРА

Архитектура јединице контролног чвора је иста као на слици 4.4 са једном малом, али веома значајном изменом. Уместо шрафираних регистара који раздвајају улазни од излазног дела процесорске јединице, у предложеној архитектури декодера постављен је FIFO бафер релативно мале дубине. За случај када се декодују регуларни кодови, или барем кодови код којих је тежина свих врста иста, не постоје разлике у функционалности. Међутим, код нерегуларних кодова, код којих сваки слој има различит број циркуланата, постоји значајна разлика. Наиме, у случају да су улазни и излазни део процесора контролног чвора

распрегнути само једним степеном регистара, потребно је уводити циклусе паузе између процесирања слојева који имају различит број циркуланата, како би се разрешили хазарди података услед превременог уписа. Ова ситуација настаје када се процесира слој који има мали број циркуланата након слоја који има велики број циркуланата. Примери су приказани на слици 4.14. У току израчунавања минимума за један слој контролне матрице, израчунавају се поруке контролних чворова за претходни слој. Ако је претходни слој имао већи број циркуланата, онда је потребно више циклуса такта да излазни део процесора генерише све поруке, него што је потребно улазном делу процесора да израчуна минимуме. Ако би улазни део процесора кренуо на процесирање наредног, трећег, слоја и уписао међурезултате из другог слоја у регистре пре него што је излазни део процесора израчунао све поруке претходног, првог, слоја, десио би се хазард података. Другим речима, излазни део процесора не би могао да генерише преостале поруке, јер су међурезултати у регистрима за распрезање промењени. Због тога је потребно увести циклусе паузе, којих је утолико више уколико је већа разлика између тежина контролних чворова два суседна слоја. У 5G NR стандарду, ова разлика може бити и $\Delta\rho = \rho_{\max} - \rho_{\min} = 19 - 3 = 16$, па се у том случају мора сачекати 16 циклуса такта, што је изузетно велики број. Обрнута ситуација настаје ако улазни део процесора обрађује слој код кога је тежина контролних чворова већа него тежина контролних чворова за које се генеришу поруке на излазу. Тада излазни део мора да сачека да минимуми и остали међурезултати буду спремни, како би наставио са генерисањем порука за наредни слој.



Слика 4.14. Временски дијаграми понашања процесора контролног чвора са и без FIFO бафера за распрезање улазног и излазног дела процесора [49]

У случају када се уместо једног степена регистара користи FIFO бафер, улазни део процесора не мора да чека да се заврши генерисање порука из претходног слоја. Међурезултати се могу уписати у бафер чим су спремни, а излазни део процесора ће их прочитати када буде кренуо у генерисање порука за наредни слој. Ово је регулисано сигналима *wrEnable* (упис) и *getNew* (читање) са слике 4.14. Поред ових сигнала, на слици је приказан и временски дијаграм низа података који говори о томе који су међурезултати уписани у FIFO бафер и који се тренутно налазе на његовом излазу. За овај једноставан

пример тај низ има свега две могуће вредности, што је и случај са кодовима из Wi-Fi, WiMAX и DVB-S2(x) стандарда. За кодове код којих је нерегуларност више изражена, дубина FIFO бафера би морала бити већа, али ни у ком случају не већа од $\lceil \rho_{\max} / \rho_{\min} \rceil$. Коришћењем FIFO бафера за распрезање улазног и излазног дела јединице контролног чвора постиже се потпуно разрешење хазарда података услед превременог уписа, па се и у потпуности укида потреба за циклусима паузе. Једини захтев који треба испоштовати је да први слој који се процесира буде један од оних који имају највећи број циркуланата.

4.3.3. МОДУЛ ЗА ТЕРМИНАЦИЈУ ДЕКОДОВАЊА

Детекција успешног завршетка декодовања и заустављање рада декодера пре него што се заврше све итерације може допринети смањеној потрошњи енергије. У региону ниске вероватноће грешке у просеку је потребно значајно мање итерација за конвергенцију у валидну кодну реч, него што је максимално предвиђено, па нема потребе за извршавањем свих унапред дефинисаних итерација. У ситуацијама када се декодовање прекида на основу сазнања да је синдром једнак нула-вектору, заустављање рада декодера се назива терминацијом (енгл. *termination*). Ако се пак декодовање прекида на основу претпоставке да се десила конвергенција у валидну кодну реч, која је исправна са одређеном вероватноћом, онда се заустављање назива раном терминацијом (енгл. *Early Termination, ET*). У литератури се може пронаћи неколико различитих метода за рану терминацију или терминацију декодовања [137]–[139].

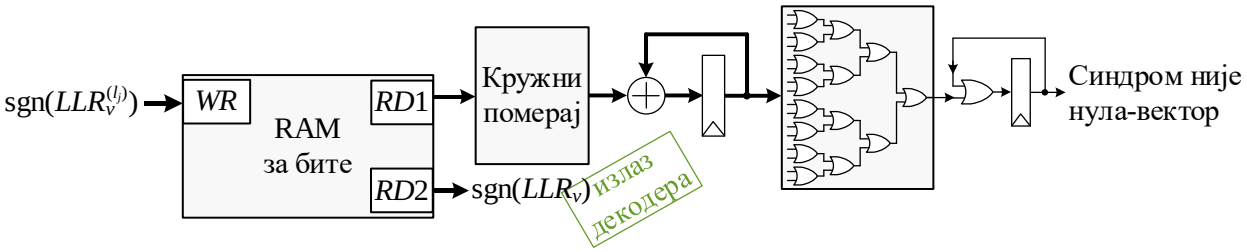
Најједноставнији начин за рану терминацију декодовања је поређење тврдих одлука кодне речи у две сукцесивне итерације [137]. Ако су бити кодне речи у две суседне итерације једнаке, претпоставља се да се они даље неће мењати и да је декодер исконвергирао у кодну реч, па се на основу тога зауставља декодовање. Међутим, нема експлицитне провере синдрома, па не постоји никаква гаранција да је донета одлука о заустављању декодовања тачна. Могуће је урадити модификацију при којој се поред наведеног критеријума посматра да ли су LLR вредности по апсолутној вредности веће од предефинисаног прага. Ако јесу, сматра се да је њихова поузданост висока, па је вероватноћа да је декодер успешно завршио декодовање већа. Ипак, овај метод, иако једноставан, ствара губитке у перформансама контроле грешака, јер се не може са сигурношћу тврдити да није требало дозволити наставак декодовања.

Провера синдрома при слојевитом декодовању мора се радити по слојевима, па је други метод за рану терминацију испитивање да ли су једначине за провере парности задовољене, на основу тренутних тврдих одлука о битима кодне речи [138]. Међутим, и овај метод изазива губитке у перформансама контроле грешака, јер се може догодити да, због слојевите природе декодовања, бити на истим позицијама имају различите вредности у различитим слојевима. У таквој ситуацији, синдром сигурно није једнак нула-вектору, јер су неке једначине провера парности провераване на основу привремених, а не финалних вредности које се прослеђују кориснику. Због тога се најчешће чека да провере парности извршене на овај начин буду задовољене у две или више сукцесивних итерација, чиме се повећава вероватноћа за исправну терминацију.

Како би се омогућило рачунање свих елемената синдрома над истим битима кодне речи, у [139] је предложено да се израчунавања у току текуће итерације раде над финалним вредностима добијеним на крају претходне итерације. Овај метод примењен је и у овој дисертацији. Код слојевитог декодовања, читање бита из претходне итерације је синхронизовано са читањем LLR вредности из текуће итерације. Адресе са којих се чита су исте, а једначине провере парности се рачунају кружним померањем одговарајућих група бита и рачунањем „ексили" операција. Код хибридног декодовања, читање LLR вредности са неких локација за наредну итерацију може да почне и пре уписа финалних вредности из

претходне итерације. Због тога је неопходно проверу синдрома за бите добијене на крају претходне итерације радити синхронизовано са уписом LLR вредности у текућој итерацији, а не са њиховим читањем.

На слици 4.15 приказана је принципска шема модула за детекцију успешне конвергенције у кодну реч. Бити над којима се врши провера уписују се у RAM меморију у току уписа LLR вредности у меморију за декодовање. Бити представљају водеће бите LLR вредности (бите знака). Меморијски блок је реализован помоћу две одвојене меморије како би се обезбедило двоструко баферисање. Једна меморија служи за проверу синдрома текуће кодне речи, док се из друге меморије читају бити претходне кодне речи ако је претходном провером синдрома детектовано да је она валидна. Меморије мењају улоге на крају декодовања сваке кодне речи. Израчунавање свих провера парности из једног слоја се врши паралелно, а на крају се проверава да ли су све вредности „ексили" операција једнаке нули у посебном стаблу „или" кола. Резултат је логичка нула ако су све једначине из слоја задовољене. На крају, та вредност се акумулира („или" операцијом) како би се проверило да ли су све једначине провера парности из свих слојева задовољене.



Слика 4.15. Модул за израчунавање синдрома и генерисање сигнала за прекид декодовања

4.4. ОПТИМИЗАЦИЈА ПЕРФОРМАНСИ ДЕКОДОВАЊА

Хибридно декодовање је у основи слојевито декодовање, али у конфликтним ситуацијама делимично прелази на симултано декодовање. Због тога, гледано у броју итерација, конвергенција алгоритма може бити спорија него код слојевитог декодовања. Еквивалентно, за исти максималан број итерација може се појавити одређени губитак у перформансама контроле грешака. Губитак је изражен када је број конфликтних ситуација велики, што се дешава код декодера који има велику дубину проточне обраде и за кодове чија је матрица основног графа густа. Велика густина матрице основног графа значи да у суседним слојевима контролне матрице постоји велики број циркуланата који одговарају истим групама варијабилних чворова. С обзиром на то да се висока учестаност сигнала такта може остварити интензивним скраћењем критичне путање и увођењем великог броја степени проточне обраде, губитак у перформансама контроле грешака се не може избећи.

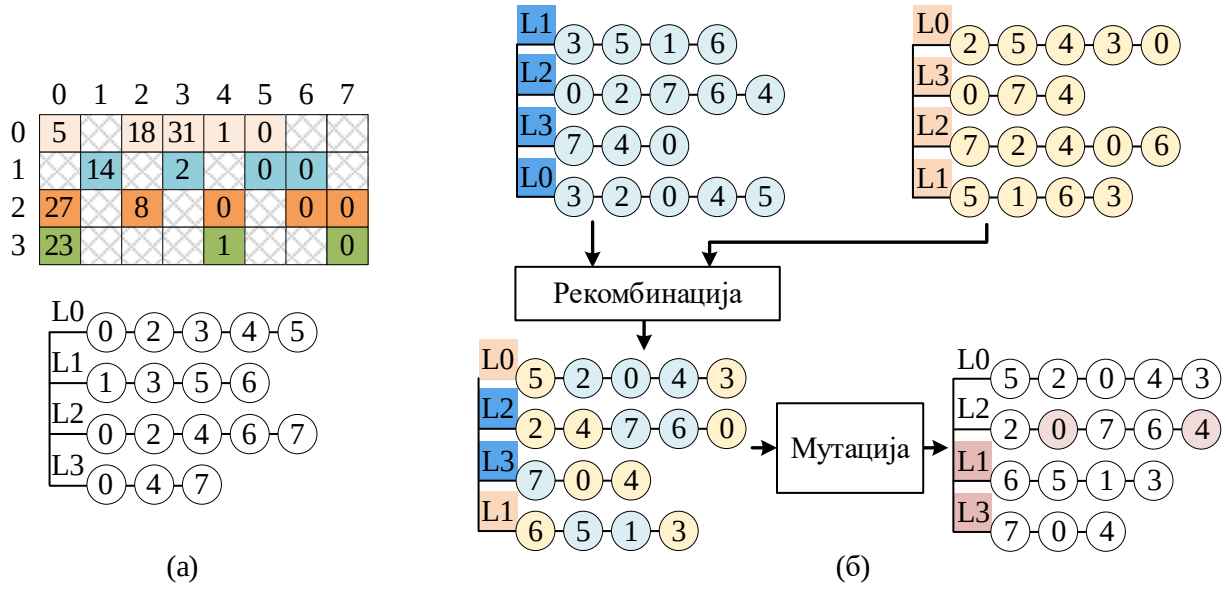
Један начин да се губитак надокнади је декодовање на већем броју итерација. Очекивано, повећањем броја итерација декодовања смањује се проток, али ако је добитак интензивног увођења проточне обраде такав да декодер може да ради на значајно вишој учестаности сигнала такта, постоји могућност да се број итерација у некој мери може повећати, а да се задржи велики проток. Другим речима, могуће је да је добитак од увођења великог броја степени проточне обраде израженији него недостатак у перформансама контроле грешака који због њих настаје.

Међутим, рад са већим бројем итерација није неопходан ако се број конфликтних ситуација смањи коришћењем измењеног редоследа процесирања контролне матрице. У дисертацији је предложен један метод за оптимизацију редоследа процесирања, којим се побољшавају перформансе контроле грешака, заснован на генетичком алгоритму. Метод је сличан оптимизацији редоследа процесирања при кодовању, описаној у одељку 3.2.2, али је

оптимизациона функција потпуно другачија, а рекомбинација и мутација се раде на делимично измењен начин.

На слици 4.6 показано је да се процесирање циркуланата у оквиру једног слоја контролне матрице може извршавати по произвољном редоследу. Тиме се добијају идентични резултати, али је смањен број циклуса паузе које је потребно увести да би се избегли хазарди података. Међутим, на перформансе контроле грешака слојевитог декодовања не утиче ни редослед процесирања слојева. LLR вредности добијене на крају декодовања се свакако разликују, али с обзиром на то да је измена редоследа процесирања еквивалентна замени места одређених врста у контролној матрици, чиме се добија еквивалентан код, перформансе контроле грешака остају неизмењене. Измена редоследа процесирања слојева коришћењем генетичког алгоритма раније је коришћена за смањење броја циклуса паузе при слојевитом декодовању [48]. Међутим, у дисертацији је предложена оптимизација која користи генетички алгоритам за проналазак оптималног редоследа процесирања слојева и, додатно, циркуланата унутар слојева. Поред тога, за разлику од метода коришћених за слојевито декодовање, код којих конфликти нису дозвољени, предложени метод оптимизације дозвољава конфликте, јер се хибридним декодовањем они могу избећи, али се оптимизацијом њихов број минимизује.

Редослед процесирања циркуланата може се представити низом од m_b вектора. Сваки вектор представља један слој у контролној матрици. Елементи вектора су позиције циркуланата, тј. индекси група варијабилних чворова. Уједно, ти индекси су и адресе којима се приступа при читању и ажурирању потребних LLR вредности. Графичка репрезентација оригиналног редоследа процесирања, који одговара једном примеру пермутационе матрице, приказана је на слици 4.16.а). Оваква представа редоследа процесирања користи се при оптимизацији.



Слика 4.16. Пример (а) пермутационе матрице и одговарајућег оригиналног редоследа процесирања и (б) рекомбинације и мутације редоследа процесирања током оптимизације генетичким алгоритмом [49]

Иницијална популација јединки генерише се из оригиналног редоследа случајним пермутацијама позиција вектора унутар низа (измена редоследа процесирања слојева) и случајним пермутацијама елемената унутар појединачних вектора (измена редоследа процесирања циркуланата унутар слојева). Селекција се врши одабиром јединки које представљају редослед којим се добија најмањи број читања неажурних LLR вредности. Треба рећи да је број читања неажурних вредности еквивалентан броју двоструких уписа и

да зависи од редоследа процесирања и од броја степени проточне обраде. Укрштање (рекомбинација) јединки за наредну генерацију се врши у два корака. Први корак је рекомбинација низова који представљају редослед процесирања слојева. Ови низови имају елементе од 1 до m_b, а укрштају се на исти начин на који је то описано у одељку 3.2.2. Дакле, прво се из првог низа са случајно одабраних позиција селектује један његов повезан део и смести на исте позиције у резултујућем низу. Остале позиције резултујућег низа се попуњавају елементима другог низа у истом редоследу, при чему се води рачуна да се елементи не понављају. Други корак представља рекомбинацију свих вектора који представљају редослед процесирања циркуланата у оквиру једног слоја. Из распореда слојева се идентификују вектори који одговарају истим слојевима, а затим се и они укрштају према идентичној процедури. Поступак рекомбинације за два распореда процесирања који одговарају пермутационој матрици са слике 4.16.а), приказан је на слици 4.16.б). Мутација јединке се врши заменом места случајно одабраних вектора унутар низа и заменом места случајно одабраних елемената унутар вектора.

4.5. РЕЗУЛТАТИ

4.5.1. ОПТИМИЗАЦИЈА РЕДОСЛЕДА ПРОЦЕСИРАЊА

Оптимизација редоследа процесирања контролне матрице даје значајно смањење конфликтних ситуација и повећава број читања ажурних LLR вредности, чиме се понашање предложеног декодера за хибридно декодовање у великој мери приближава потпуном слојевитом декодовању. На слици 4.17 приказани су резултати оптимизације за кодове из 5G NR стандарда за све кодне количнике изведене из основног графа 1 и за различит број степени проточне обраде. Приказан је број читања ажурних LLR вредности у току једне итерације декодовања за оригиналан редослед процесирања и за оптимизовани редослед процесирања.

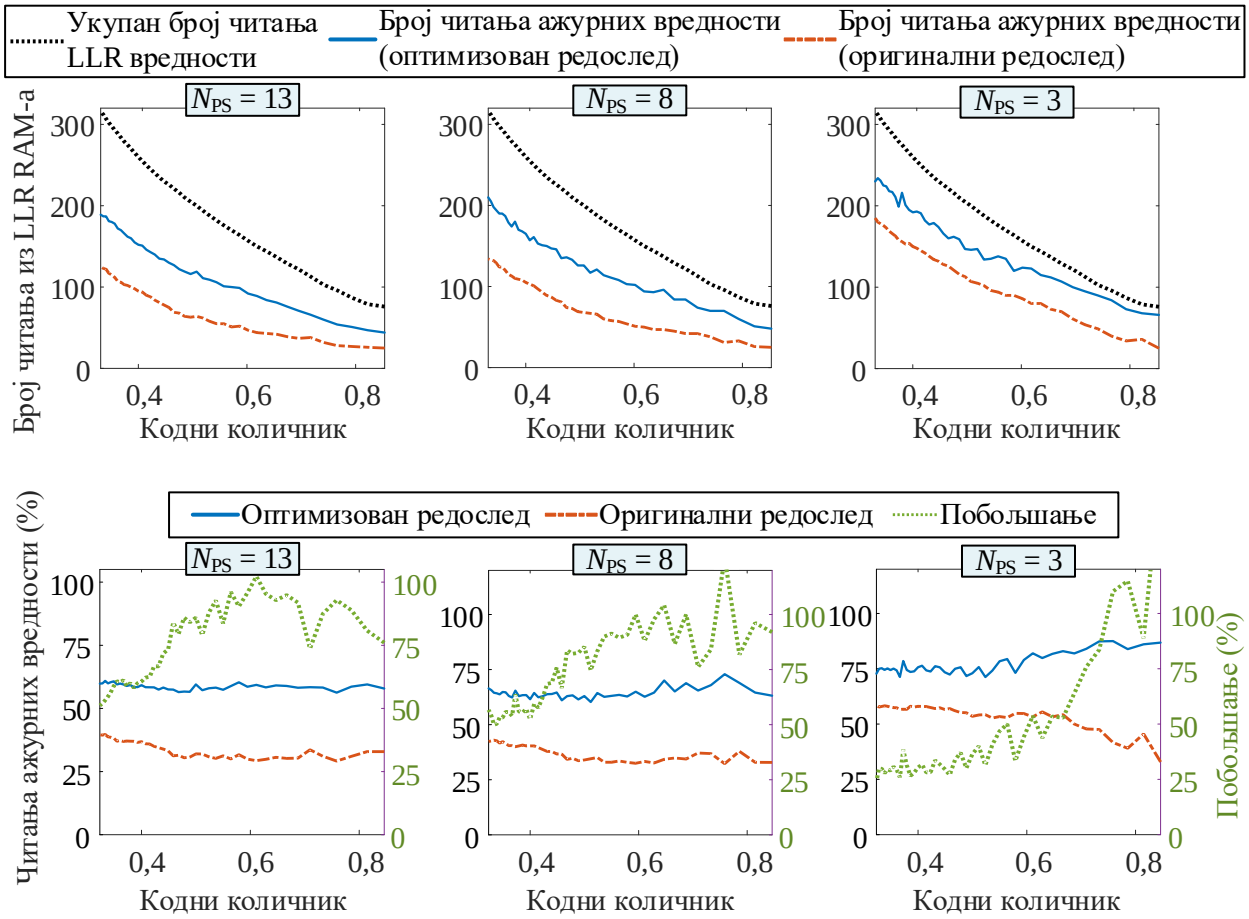
Поред апсолутних бројева, приказани су и проценти читања ажурних LLR вредности у односу на укупан број читања, као и побољшање добијено оптимизацијом изражено у процентима, израчунато према следећој формули:

$$I = \frac{N_{\text{up-to-date, opt}} - N_{\text{up-to-date, orig}}}{N_{\text{up-to-date, orig}}} \cdot 100\%, \tag{100}$$

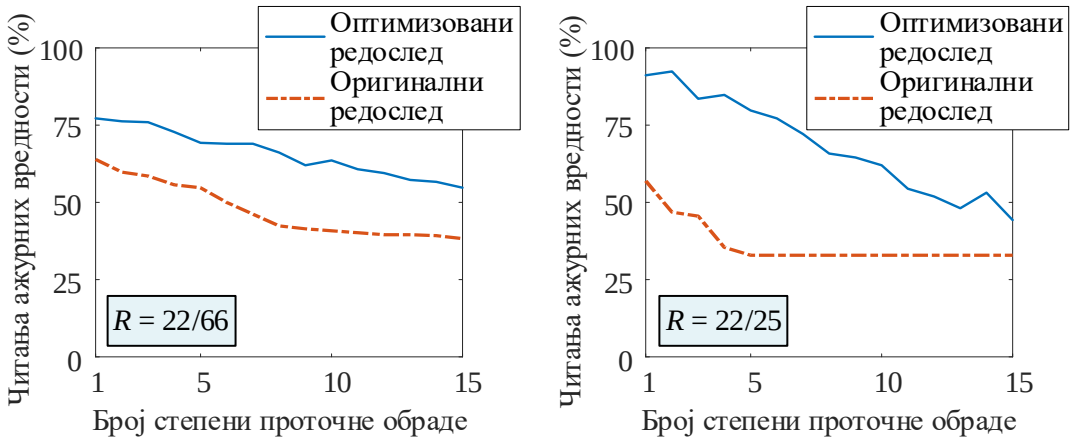
где је $N_{\text{up-to-date, opt}}$ број читања ажурних вредности при оптимизованом редоследу процесирања, а $N_{\text{up-to-date, orig}}$ при оригиналном редоследу процесирања. Остварена побољшања су између 25% и 120%, зависно од кодног количника и броја степени проточне обраде. Може се приметити да је побољшање у просеку веће за велике кодне количнике него за мале кодне количнике. Ово је последица тога да су пермутационе матрице кодова са великим кодним количницима у 5G NR стандарду гушће него код кодова са малим кодним количницима. Због тога је број ажурних читања мањи. Иако је побољшање за веће кодне количнике веће, не значи да је њихов крајњи губитак у перформансама контроле грешака мањи.

На слици 4.18 приказан је проценат читања ажурних вредности у зависности од броја степени проточне обраде за један 5G NR код малог кодног количника ($R = 22/66$) и један код великог кодног количника ($R = 22/25$). Оба кода су изведена из основног графа 1, при чему је узето у обзир пунктирање првих $2Z$ информационих бита. Резултати важе за све величине циркуланата, јер редослед процесирања не зависи од величине циркуланта, већ само од њихових позиција у контролној матрици. Са слике се може уочити јасан пад у броју

ажурних читања како број степени проточне обраде расте. Додатно, овај пад је израженији за велики кодни количник, што је и очекивано због густине пермутационе матрице. Стога се може очекивати да ће губитак у перформансама контроле грешака за велики број степени проточне обраде бити већи ако се декодују кодови чији је кодни количник већи.



Слика 4.17. Резултати оптимизације редоследа процесирања генетичким алгоритмом у апсолутним бројевима и у процентима за 5G NR кодове изведене из основног графа 1 и за различит број степени проточне обраде N_{PS} [49]



Слика 4.18. Проценат читања ажурних LLR вредности у функцији од броја степени проточне обраде за 5G NR кодове изведене из графа 1 кодног количника 22/66 и 22/25 [49]

4.5.2. ЗДРУЖЕНА АНАЛИЗА ПЕРФОРМАНСИ КОНТРОЛЕ ГРЕШАКА И ПРОТОКА

Како би се детаљно испитале перформансе контроле грешака предложеног алгоритма декодовања, извршене су опсежне Монте Карло симулације. Реализован је софтверски модел декодера који користи Min-Sum алгоритам са офсетом, код кога су све поруке и LLR вредности представљене у формату са фиксним зарезом. LLR вредности су представљене помоћу осам бита, а све поруке помоћу шест бита. Сви међурезултати су представљени помоћу минималног броја бита потребног да се спрече прекорачења при сабирању, а за поруке и LLR вредности је уведено засићење на максималну или минималну вредност.

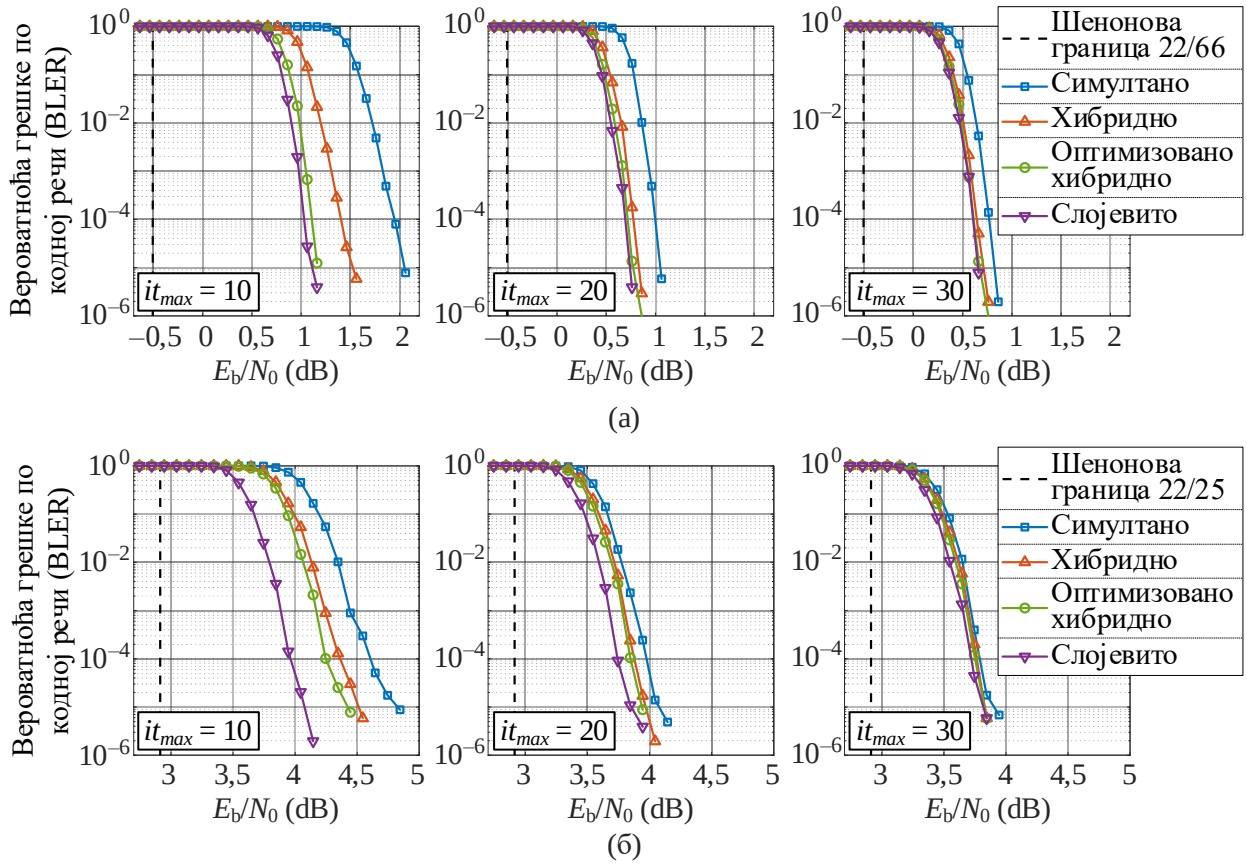
Описани софтверски модел у потпуности представља модел реализованог хардвера и има исте перформансе контроле грешака као и декодер реализован на FPGA чипу. Иако су перформансе контроле грешака потврђене и за хардверску реализацију, софтверске Монте Карло симулације омогућавају поређења са декодером који реализује слојевито декодовање и декодером који реализује симултано декодовање. За декодер који реализује хибридно декодовање, симулације су извршене и за оригинални и за оптимизовани редослед процесирања. Како би се независно посматрале само перформансе декодера, у симулационом окружењу је предвиђено да се модулација, генерисање шума и демодулација раде максималном тачношћу коју подржава представа бројева са покретним зарезом, али су LLR вредности квантизоване пре иницијализације декодера. Број кодних речи који је симулиран за једну вредност односа сигнал-шум је динамички подешаван тако да се увек чека минимално десет неуспешно декодованих речи. За симулације описане у овом одељку, број симулираних кодних речи није прелазио милион, што омогућава релативно поуздану естимацију вероватноће грешке по кодној речи за вероватноће грешке веће од 10^{-5}.

Симулације су извршене за три различите вредности максималног броја итерација ($it_{\max}=10$, $it_{\max}=20$ и $it_{\max}=30$). Број степени проточне обраде је постављен на 13. Ово је број који је обезбедио учестаност рада декодера вишу од 400 MHz у реализацији на FPGA чипу, о чему је више речено у одељку 4.5.3.

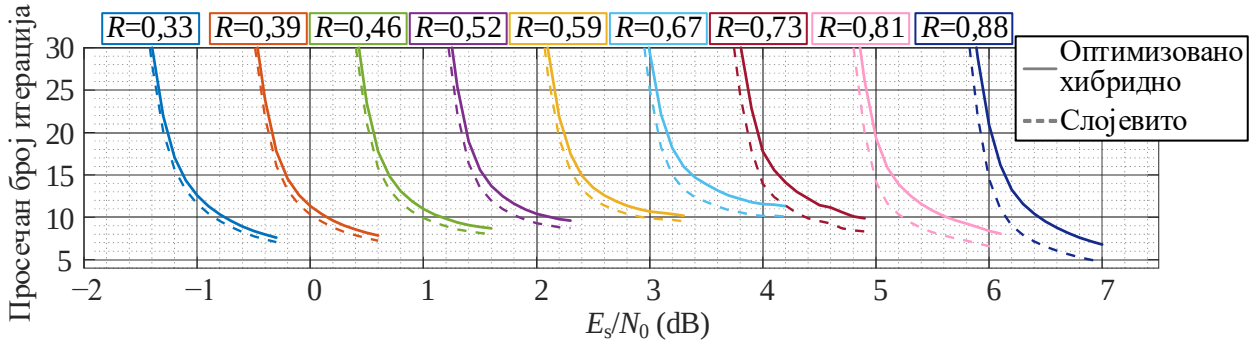
На слици 4.19 приказани су резултати симулација за кодове из 5G NR стандарда, једног малог ($R=22/66$) и једног великог ($R=22/25$) кодног количника, и то кода (25344, 8448) и кода (9600, 8448). Оба кода су изведена из основног графа 1 за највећу вредност величине циркуланта $Z=384$. С обзиром на то да у практичним корисничким сценаријима за сада није предвиђено коришћење BPSK модулације у 5G NR стандарду [140], симулације су вршене за AWGN канал и QPSK модулацију (енгл. *Quadrature Phase Shift Keying*).

Из резултата се може закључити да оптимизовано хибридно декодовање даје боље резултате него хибридно декодовање по оригиналном редоследу процесирања контролне матрице, посебно за мали кодни количник и мали максималан број итерација. Међутим, у свим ситуацијама постоји губитак у перформансама у односу на потпуно слојевито декодовање. Значајнији губитак (око 0,2 dB) је уочљив једино за већи кодни количник при малом броју итерација декодовања. Ако се дозволи већи број итерација или се користи нижи кодни количник, губитак је мањи од 0,1 dB. Ово понашање није неочекивано с обзиром на то да се хибридно декодовање повремено понаша као симултано декодовање, а губитак симултаног декодовања је значајно већи ако се декодовање врши са малим максималним бројем итерација.

Као додатак на претходну анализу перформанси, на слици 4.20 приказан је и просечан број итерација потребан за успешно декодовање за неколико различитих кодова из 5G NR стандарда. Резултати потврђују да се на вишим кодним количницима јавља већа разлика у перформансама слојевитог декодовања и оптимизованог хибридног декодовања.



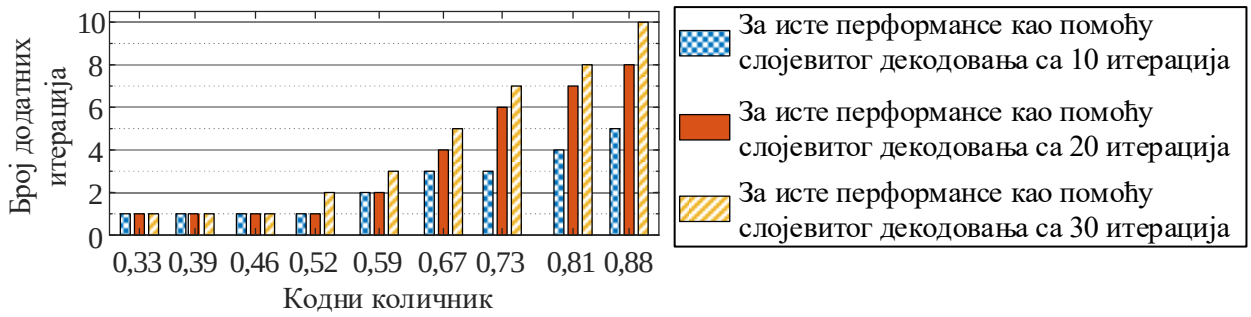
Слика 4.19. Перформансе контроле грешака за симултано, слојевито, хибридно и оптимизовано хибридно декодовање за кодове из 5G NR стандарда: (а) (25344, 8448) кодног количника R = 22/66 и (б) (9600, 8448) кодног количника R = 22/25 [49]



Слика 4.20. Просечан број итерација потребан за декодовање 5G NR кодова коришћењем слојевитог и оптимизованог хибридног декодовања. Симулирани су кодови (слева надесно): (25344, 8448), (21504, 8448), (18432, 8448), (16128, 8448), (14208, 8448), (12672, 8448), (11520, 8448), (10368, 8448) и (9600, 8448) [49].

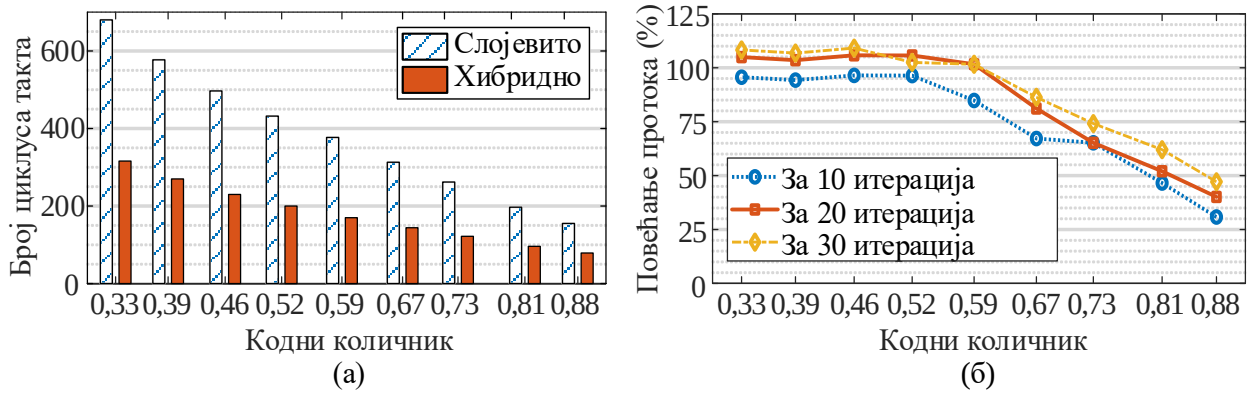
Коначно, испитано је колико је потребно повећати максималан број итерација декодера који реализује хибридно декодовање да би се добиле исте перформансе контроле грешака као за декодер који реализује слојевито декодовање. На слици 4.21 приказани су резултати. Они потврђују претходне резултате и показују да је за мале кодне количнике довољно додати свега једну итерацију декодовања за постизање истих или бољих перформанси контроле грешака, док је за више кодне количнике потребан већи број додатних итерација. Најлошији резултат је за кодни количник 0,88 (22 / 25) при малом броју итерација. Тада је потребно 15 итерација за постизање истих перформанси као код слојевитог декодовања на 10 итерација,

што је повећање од 50%. Међутим, чак и у том случају се за исте перформансе контроле грешака може добити повећање у протоку.



Слика 4.21. Потребан број додатних итерација оптимизованог хибридног декодера за исте перформансе контроле грешака као код слојевитог декодовања (исто E_b/N_0 за BLER 10^{-5}) [49]

Како би се упоредио проток предложеног декодера и декодера који користи слојевито декодовање, неопходно је прво одредити потребан број циклуса такта по једној итерацији. С обзиром на то да су сви циклуси паузе уклоњени, број потребних циклуса такта по итерацији за хибридно декодовање једнак је броју циркуланата у контролној матрици кода. Међутим, код слојевитог декодовања је неопходно увести циклусе паузе када год настану конфликтне ситуације. На слици 4.22.а) приказан је број потребних циклуса такта по итерацији за неколико различитих кодова из 5G NR стандарда (исти кодови као на слици 4.20). Како би поређење било коректно, при одређивању броја циклуса за слојевито декодовање, сматрано је да не постоје додатни циклуси паузе због хазарда података у процесорима контролних чворова, тј. подразумевано је да су хазарди разрешени уметањем FIFO бафера за распрезање улазног и излазног дела процесора. Такође, број циклуса је израчунат за оптимизовани редослед процесирања. За сваки посматрани код урађена је оптимизација редоследа процесирања на исти начин као за хибридно декодовање, са разликом што је оптимизациона функција за слојевито декодовање написана да даје број циклуса паузе. Циљ оптимизације је проналажење редоследа процесирања који даје најмањи број циклуса паузе. Међутим, и поред наведених разматрања, број циклуса паузе потребних за класично слојевито декодовање и даље је изузетно велики.



Слика 4.22. (а) Потребан број циклуса такта за једну итерацију декодовања за декодере који реализују слојевито и хибридно декодовање за 13 степени проточне обраде и (б) остварено повећање протока декодера за хибридно декодовање у односу на декодер за слојевито декодовање за исте перформансе контроле грешака [49]

На основу резултата са слике 4.22.а) израчунат је информациони проток за оба декодера при чему је број итерација декодера за хибридно декодовање подешен тако да перформансе контроле грешака буду исте као код слојевитог декодовања са максималних 10,

20 и 30 итерација. Проток је израчунат за декодер са 13 степени проточне обраде. Извршено је поређење добијених вредности и израчунато побољшање које се добија хибридним декодовањем у односу на слојевито декодовање. Резултати су приказани на слици 4.22.б). Остварено побољшање је изузетно велико и креће се од 30,8%, за кодове високог кодног количника при малом броју итерација, до 109,1%, за кодове ниског кодног количника при великом броју итерација.

Овим је показана супериорност предложеног метода декодовања у односу на слојевито декодовање за кодове из 5G NR стандарда. Ако се разматра архитектура декодера за слојевито декодовање која користи само једну мрежу за кружни померај (архитектура са слике 4.7), број циклуса паузе може бити мањи него на слици 4.22.а). Разлог за то је краћа критична путања, па се иста максимална учестаност рада може постићи и са мањим бројем степени проточне обраде. У дисертацији је мрежа за кружни померај пројектована тако да има четири степена проточне обраде, па би стога архитектура са слике 4.7 могла бити реализована са девет степени проточне обраде. Такав декодер је предложен у раду [132]. Међутим, предложени декодер са 13 степени проточне обраде и тада остварује побољшање у протоку између 16,5% и 75,2%. Притом нису урачунати циклуси такта потребни за инверзну пермутацију LLR вредности које су након завршеног декодовања остале уписане у меморију у измењеном редоследу.

4.5.3. ЕФИКАСНОСТ ИСКОРИШЋЕЊА ХАРДВЕРСКИХ РЕСУРСА

Описани декодер за хибридно декодовање имплементиран је на развојном систему Xilinx ZCU111 са Zynq Ultrascale+ RF-SoC чипом (XCZU28DR) и то за WiMAX, DVB-S2(x) и 5G NR стандарде. Међутим, декодер је флексибилан и може да декодује било који квазицикличан код, докле год је величина циркуланта подржана мрежом за кружни померај. Број процесорских јединица контролних и варијабилних чворова у имплементираним верзијама декодера је подешен тако да одговара највећој величини циркуланта (96 за WiMAX, 360 за DVB-S2(x) и 384 за 5G NR), а све три верзије подржавају све кодове дефинисане одговарајућим стандардом.

Број степени проточне обраде је 13 у декодерима за WiMAX и 5G NR стандард, а 11 у декодеру за DVB-S2(x), јер је нефлексибилна мрежа за кружни померај у DVB-S2(x) декодеру мање комплексна. Остварене максималне учестаности рада су 585 MHz за WiMAX, 373 MHz за DVB-S2(x) и 404 MHz за 5G NR декодер. За највећу величину циркуланата и за максимално 10 итерација декодовања за све стандарде су остварени информациони протоци од више гигабита у секунди, а детаљни резултати за неколико одабраних кодова приказани су у табели 4.1. Поред информационог протока, израчунат је и кодни проток као

$$T_{\text{cod}} = \left(n \cdot f_{\text{CLK}}\right) / \left(N_{\text{circ}} \cdot it_{\max}\right). \tag{101}$$

Кашњење декодера (l_{d}) одређено је као време од последњег трансфера LLR вредности за једну кодну реч на улазу у декодер до последњег трансфера декодованих LLR вредности за ту исту кодну реч на излазу из декодера. Ширина података на интерфејсима је 128 бита, тј. 16 LLR вредности, па и она у значајној мери утиче на кашњење декодера. Кориснику би од интереса могли бити само информациони бити, па би у том случају кашњење било значајно мање. Ипак, могуће је да се у другој врсти канала декодер користи на другачији начин [141], па је у прорачуну за кашњење сматрано да се читају целе LLR вредности комплетне кодне речи.

Треба нагласити да је у свим досадашњим анализама протока сматрано да декодер мора да ради у реалном времену, због чега је у свим прорачунима одабрано дељење са максималним бројем итерација. Међутим, када је вероватноћа грешке по кодној речи мала, за највећи број кодних речи декодовање се заврши у некој од итерација која није последња.

Стога би се уместо максималног броја итерација могао користити и просечан број итерација, чиме би се добили значајно већи протоци. Ова метрика је од интереса у ситуацији када се експлицитно не захтева рад у реалном времену са строгим временским оквирима за завршетак декодовања. Такве ситуације се јављају у већини корисничких сценарија потрошачке електронике (енгл. *consumer electronics*).

Табела 4.1. Остварени проток декодера за максималних 10 итерација декодовања

Стандард	Дужина кодне речи	Кодни количник	Број циркуланата	$T_{\text{norm,info}}$ (b/cycle)	$T_{\text{norm,cod}}$ (b/cycle)	$f_{\max}$ (MHz)	T_{info} (Gb/s)	T_{cod} (Gb/s)	l_{d} (μs)
WiMAX	2304	1/2	76	1,52	3,03	585	0,89	1,77	1,54
	2304	2/3A	81	1,90	2,84		1,11	1,66	1,63
	2304	3/4A	85	2,03	2,71		1,19	1,59	1,70
DVB-S2x	64800	2/9	560	2,57	11,57	373	0,96	4,32	25,84
	64800	90/180	710	4,56	9,13		1,70	3,41	29,85
	64800	20/30	781	5,53	8,30		2,07	3,10	31,75
	64800	140/180	785	6,42	8,25		2,40	3,08	31,86
	64800	154/180	759	7,30	8,54		2,73	3,19	31,16
5G NR BG1	25344	22/66	316	2,67	8,02	404	1,08	3,25	11,84
	16896	22/44	210	4,02	8,05		1,63	3,26	7,92
	12672	22/33	144	5,87	8,80		2,37	3,56	5,63
	10368	22/27	96	8,80	10,80		3,56	4,37	4,09
	9600	22/25	79	10,69	12,15		4,33	4,92	3,55
5G NR BG2	19200	10/50	197	1,95	9,75		0,79	3,95	7,95
	11520	10/30	121	3,17	9,52		1,28	3,85	4,89
	7680	10/20	77	4,99	9,97		2,02	4,04	3,21
	5760	10/15	52	7,38	11,08		2,99	4,48	2,29

$T_{\text{norm,info}}$: нормализовани информациони проток у битима по циклусу такта; $T_{\text{norm,cod}}$: нормализовани кодни проток у битима по циклусу такта одређен са $T_{\text{norm,info}}/R$; T_{info}: остварени информациони проток; T_{cod}: остварени кодни проток; l_{d}: кашњење декодера

Резултати заузећа хардверских ресурса за сва три декодера приказани су у табели 4.2, а израчунате су и ефикасности искоришћења хардверских ресурса. Поред тога, дато је поређење са релевантним резултатима из литературе. Поређења су извршена са следећим реализацијама декодера: декодерима за WiMAX стандард из [56], [133] и [134], декодерима за DVB-S2x стандард из [56] и [132], као и недавно публикованом реализацијом декодера за 5G NR стандард [142] и једним комерцијалним решењем компаније Xilinx [46]. Узимајући у обзир резултате приказане у референтним радовима, резултати за проток су дати за најдужи WiMAX кôд кодног количника 3/4A, дугачки DVB-S2x кôд кодног количника 140/180 и за најдуже 5G NR кодове изведене из основног графа 1 кодних количника 22/66 и 22/25. Како би се направило униформно поређење, протоци дати у табели су израчунати за једну итерацију декодовања.

Сви FPGA чипови коришћени за реализације приказане у табели 4.2 имају лукап табеле које могу да реализују логичке функције са шест улаза. Међутим, декодер предложен у дисертацији тестиран је на FPGA чипу најновије Ultrascale+ генерације који је израђен у 14-нанометарском технолошком процесу. Како је већина FPGA чипова коришћених у

осталим радовима израђена у технолошком процесу са дужином канала транзистора од 28 nm, предложени декодер је имплементиран и за један такав чип, конкретно чип из серије Virtex-7 (XC7VX690T). Ово је урађено како би се показало да су предности у односу на радове из литературе последица архитектуралних и алгоритамских решења, а не коришћења FPGA чипа из модерније фамилије, што је препорука дата у релевантној студији [143].

Табела 4.2. Резултати имплементације декодера за WiMAX, DVB-S2(x) и 5G NR стандарде и поређење са референтним радовима из литературе [49]

	Стандард	n	FPGA проц.	Q	Заузеће ресурса				f_{max} (MHz)	$T_{norm/it}$ (Gb/s)	Ефикасност			
					Slice	LUT	FF	BRAM			Mb/s/ kSlice	Mb/s/ kLUT	Mb/s/ kFF	Mb/s/ BRAM
[133]	WiMAX	2304	28 nm	(-,6)	3732	12250	3732	24	150	1,8*	482,3	146,9	482,3	75,0
[134]	WiMAX	2304	65 nm	(-,2)	1137	3522	847	14,5	162	0,8	703,6	227,1	**944,5**	55,2
				(-,2)	5583	18542	3992	80	126	3,2	573,2	172,6	801,6	40,0
[132]	DVB-S2x	64800	28 nm	(8,6)	-	63694	75372	-	250	8,0*	-	125,6	106,1	-
[56]	WiMAX	2304	28 nm	(-,4)	12496	40700	35013	40.5	143	**8,1**	648,2	199,0	231,3	200,0
	DVB-S2x	64800		(-,4)	59874	198810	112613	252.5	80	**23,3**	389,2	117,2	206,9	92,3
[142]	5G NR	25344	28 nm	(8,6)	-	74373	46517	198,5	160	2,7	-	36,3	58,0	13,6
		9600		(8,6)						9,7	-	130,4	208,5	48,9
	5G NR	25344	14 nm	(8,6)	-	71731	46474	198,5	225	3,8	-	53,0	81,8	19,1
		9600		(8,6)						13,6	-	189,6	292,6	68,5
[46]♠	5G NR	25344	28 nm	(8,6)	-	54731	56998	119	220	1,4	-	25,6	24,6	11,8
		9600		(8,6)						7,2	-	131,6	126,3	60,5
Ова дисертација	WiMAX	2304	28 nm	(8,6)	7906	24228	23290	33,5	314	6,4	**809,5**	**264,2**	274,8	**191,0**
	DVB-S2x	64800		(8,6)	23475	73136	68795	157,5	287	18,4	**783,8**	**251,6**	**267,5**	**116,8**
	5G NR	25344		(8,6)	30824	100929	85431	136,5	261	**6,9**	**223,9**	**68,4**	**80,8**	**50,5**
		9600		(8,6)						**27,9**	**905,1**	**276,4**	**326,6**	**204,4**
	WiMAX	2304	14 nm	(8,6)	4388	24043	23592	33,5	585	11,9	2711,9	494,9	504,4	355,2
	DVB-S2x	64800		(8,6)	13762	69274	69940	157,5	373	24,0	1743,9	346,5	343,2	152,4
	5G NR	25344		(8,6)	17665	96625	87751	136,5	404	10,8	611,4	111,8	123,1	79,1
		9600		(8,6)						43,3	2451,2	448,1	493,4	317,2

n: дужина кодне речи; FPGA проц: технолошки процес у коме је израђен FPGA чип; Q: квантизациона шема, (L,M) има значење да се користи L бита за LLR вредности и M бита за поруке; f_{max}: максимална учестаност сигнала такта; $T_{norm/it}$: информациони проток нормализован на једну итерацију декодовања; Ефикасност: информациони проток $T_{norm/it}$ подељен са бројем употребљених ресурса; -: недоступан податак; *: израчунато на основу формуле дате у референци; ♠: комерцијално решење, подаци преузети из [142]. Подебљане вредности представљају најбољи резултат. У поређење нису укључени резултати за Ultrascale+ FPGA (14 nm).

Поређењем остварених резултата са претходним радовима показује се да је остварена значајно већа ефикасност искоришћења хардверских ресурса за скоро све метрике и све стандарде. Такође, остварени информациони протоци за 5G NR имају највеће вредности, док изразито паралелна архитектура из [56] остварује највеће вредности протока за WiMAX и DVB-S2x стандарде. Поред поређења остварених протока, понекад се пореде и

нормализоване вредности у битима по циклусу такта, као што је то урађено за реализацију кодера у одељку 3.4.2. Међутим, главни допринос овог дела дисертације је потпуно уклањање везе између броја степени проточне обраде и броја циклуса такта потребних за декодовање, чиме је омогућено интензивно скраћивање критичних путања и повећање учестаности рада. У другим решењима то није случај, па се мора правити компромис који даје најбољи резултат. Типичан пример је реализација 5G NR декодера из [142], која остварује максималну учестаност такта од 225 MHz на FPGA чипу из најновије фамилије. Када се уведе довољан број степени проточне обраде, као у овој дисертацији, учестаност сигнала такта може бити виша од 400 MHz. Због тога је сматрано да поређење нормализованих вредности за проток није коректно у овој ситуацији.

4.6. АНАЛИЗА И ПОБОЉШАЊЕ ПЕРФОРМАНСИ У РЕГИОНУ НИСКЕ ВЕРОВАТНОЋЕ ГРЕШКЕ

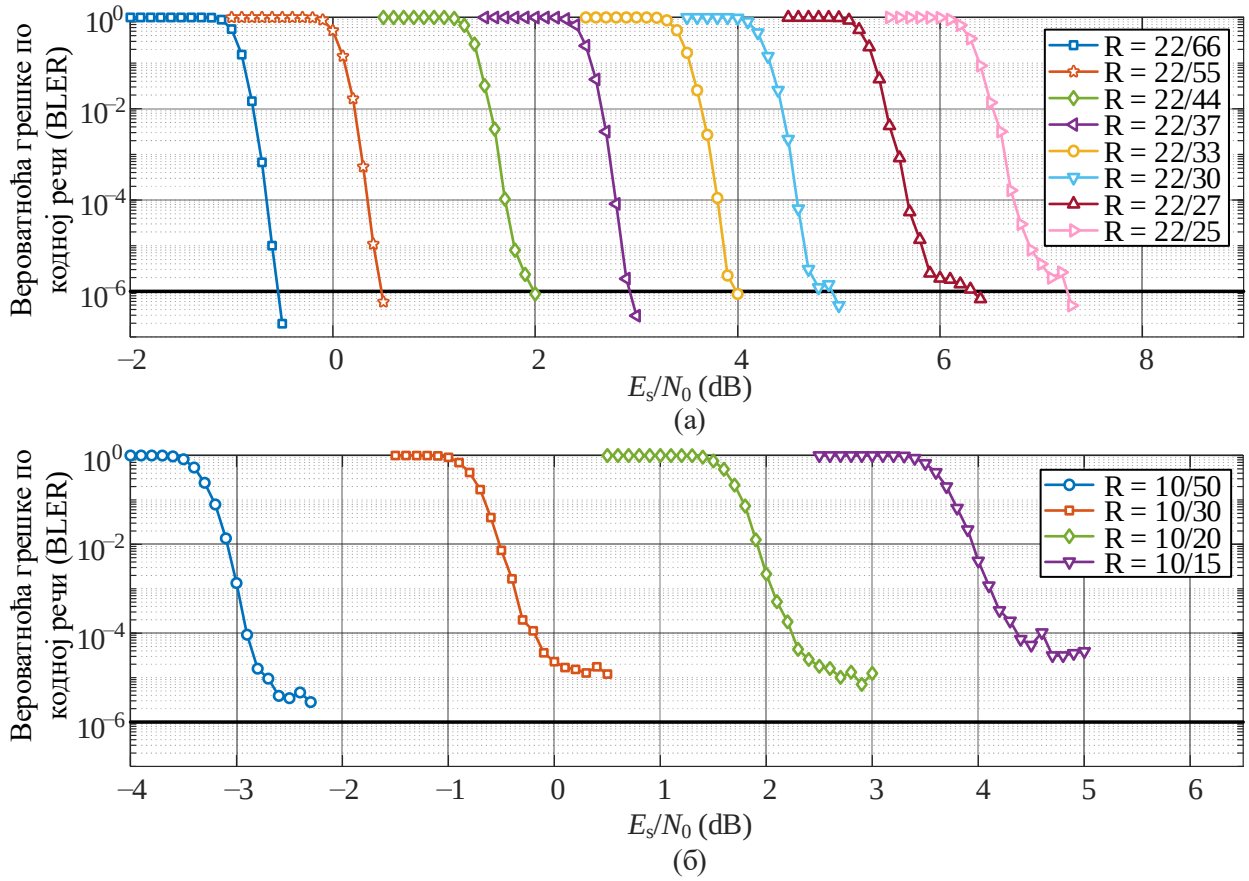
За већину потреба 5G корисничког сценарија веома поузданих комуникација малог кашњења (енгл. *Ultra-Reliable Low-Latency Communication, URLLC*), тренутни захтеви су да вероватноћа грешке по кодној речи треба да буде мања од 10^{-5}. Перформансе контроле грешака предложеног декодера управо су испитиване према овом захтеву у симулацијама чији су резултати приказани на слици 4.19. Међутим, постоје индикације да ће у одређеним применама индустријске аутоматизације бити потребна још већа поузданост [61]. Због тога су извршене додатне симулације за више различитих 5G NR кодова изведених из основног графа 1, а анализа је проширена и на кодове из основног графа 2.

На слици 4.23 приказани су резултати симулација за 10 милиона кодних речи, QPSK модулацију и AWGN канал. Симулирани су кодови чија је величина циркуланата $Z = 384$. Из резултата за кодове изведене из основног графа 1 (слика 4.23.а)) уочава се да за велике кодне количнике вероватноћа грешке по кодној речи не опада са великим нагибом за вредности испод 10^{-5}, тј. јавља се заравњење BLER криве (енгл. *error floor*). Ови ефекти су још више изражени код свих кодова изведених из основног графа 2 (слика 4.23.б)). За примене у мобилним комуникацијама (*enhanced Mobile Broadband, eMBB*) ово понашање не представља проблем, јер су захтеви да вероватноћа грешке буде испод 10^{-2}, додуше уз додатне ефекте простирања сигнала (интерференције, фединга). Повећана поузданост комуникација се у тим условима обезбеђује HARQ процедурама, тј. ретрансмисијом. Међутим, за горенаведени URLLC кориснички сценарио, вероватноћа грешке од 10^{-4}, што је најлошији резултат са слике 4.23, неприхватљива је.

Анализом LLR вредности и вредности порука установљено је да ефекат заравњења BLER криве на ниским вероватноћама грешке настаје због прераног засићења, тј. због малог динамичког опсега. Поруке су у досадашњој реализацији представљане са шест бита, а LLR вредности са осам. Проблем настаје када већина вредности нарасте до максималних, па порука, која је по апсолутној вредности мања од LLR вредности, не може да јој промени знак и декодер остаје заробљен са неколико погрешних бита који се не могу исправити. Треба нагласити да овакво понашање није карактеристично само за хибридно декодовање, већ и за потпуно слојевито декодовање. Повећањем динамичког опсега добијају се значајно боље перформансе. На пример, већ са девет бита за представу LLR вредности и са седам бита за представу порука, не уочавају се заравњења у BLER кривама за симулиране случајеве. Међутим, оваква измена захтева значајно веће заузеће хардверских ресурса.

За повећање динамичког опсега порука, уз задржавање мањег броја бита за представу, може се користити нелинеарна квантизација [63], [144]. Та идеја је искоришћена и за повећање динамичког опсега порука декодера предложеног у дисертацији. LLR вредности су представљене помоћу девет бита, али су све поруке нелинеарно квантизоване па су, уместо

помоћу седам или ранијих шест, представљене помоћу свега пет бита. Циљ је да се повећано заузеће хардверских ресурса, које настаје због повећања битске ширине за LLR вредности, надокнади мањим заузећем ресурса у блоковима који раде са порукама.

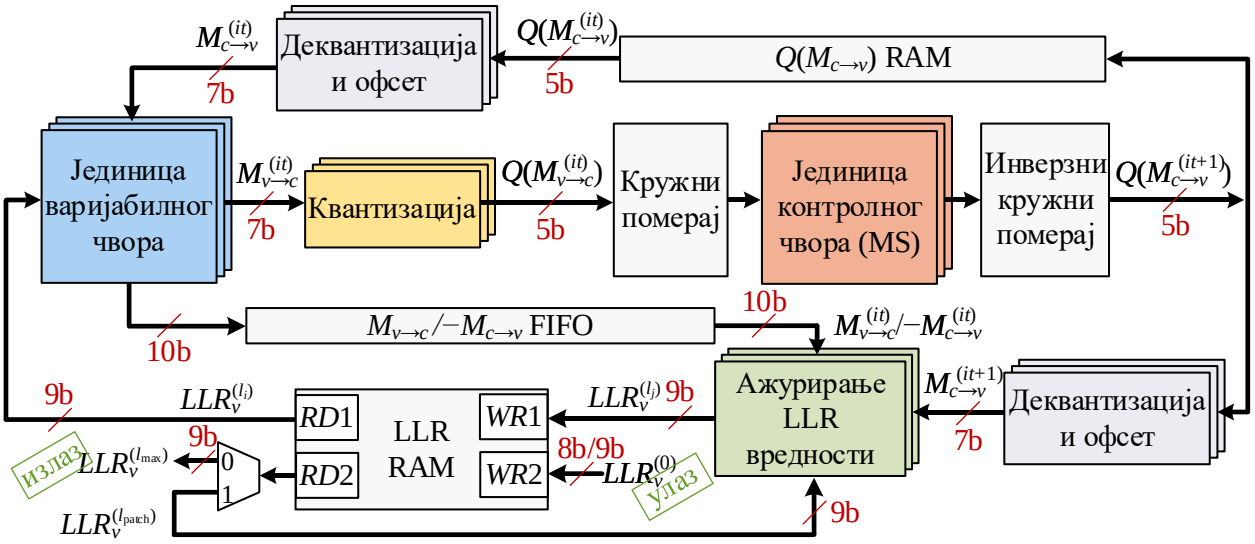


Слика 4.23. Перформансе контроле грешака декодера за хибридно декодовање и 5G NR кодове изведене из (а) основног графа 1 и (б) основног графа 2. Величина циркуланта за све кодове је 384.

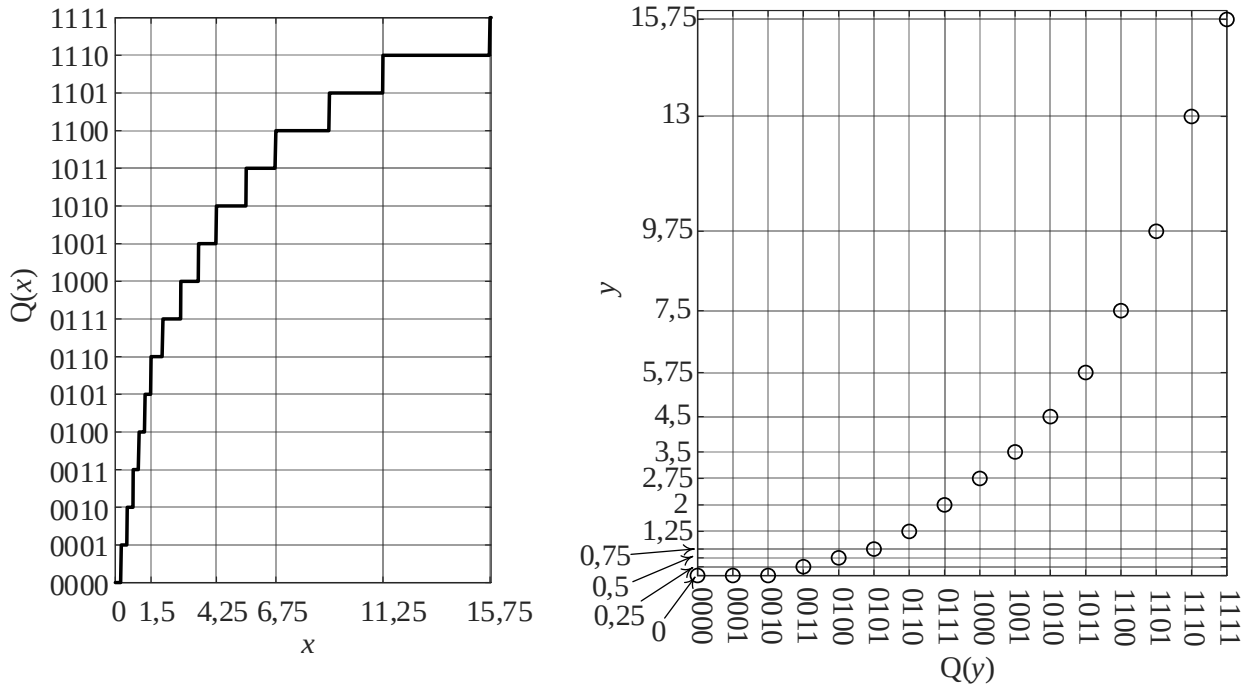
Измењена архитектура декодера за хибридно декодовање приказана је на слици 4.24. Улазне LLR вредности се могу и даље представљати помоћу 8 бита како би се уштедели ресурси у улазном интерфејсу, али се пре уписа у меморију мора урадити екстензија знака. Нелинеарна квантизација је реализована помоћу лукап табела постављених на одговарајућа места. Квантизатор је постављен пре мреже за кружни померај. Јединица контролног чвора реализује функцију Min-Sum алгоритма, тј. не одузима офсет пошто су вредности нелинеарно квантизоване, јер би одузимање офсета у линеарном домену унело велику грешку. Нелинеарно квантизоване поруке контролних чворова се у компресованом формату чувају у меморији за касније коришћење, а пре улаза у јединицу за ажурирање LLR вредности, морају се вратити у линеарно квантизоване поруке. Поред деквантизације, неопходно је применити и офсет који, ако је константан, може бити инкорпориран у вредности из лукап табеле, па ова операција не захтева додатне ресурсе. Исти блокови за деквантизацију и примену офсета морају се поставити на излазу из меморије за поруке контролних чворова, како би се добиле вредности у линеарном домену над којима се могу вршити аритметичке операције у јединицама варијабилних чворова.

Функције нелинеарног квантизатора и нелинеарног деквантизатора приказане су на слици 4.25. Функције су приказане само за апсолутне вредности порука. Оне су једино и потребне, јер је конверзија из представе негативних вредности у комплементу двојке у представу знаком и апсолутном вредношћу у новој реализацији урађена пре квантизације. Конверзија у обрнутом смеру врши се након деквантизације. На овај начин је број бита на

улазу у функције квантизатора и деквантизатора смањен за један. На слици се може уочити правило по коме су мале вредности порука квантизоване са вишом тачношћу, док су велике вредности порука квантизоване са нижом апсолутном, али сличном релативном тачношћу. Тачна представа малих порука је од суштинског значаја за перформансе контроле грешака у региону средње вероватноће грешке, тј. у региону стрмог пада BLER криве (енгл. *waterfall region*), док је за мале вредности вероватноће грешке важно да динамички опсег порука и LLR вредности буде довољно велики.



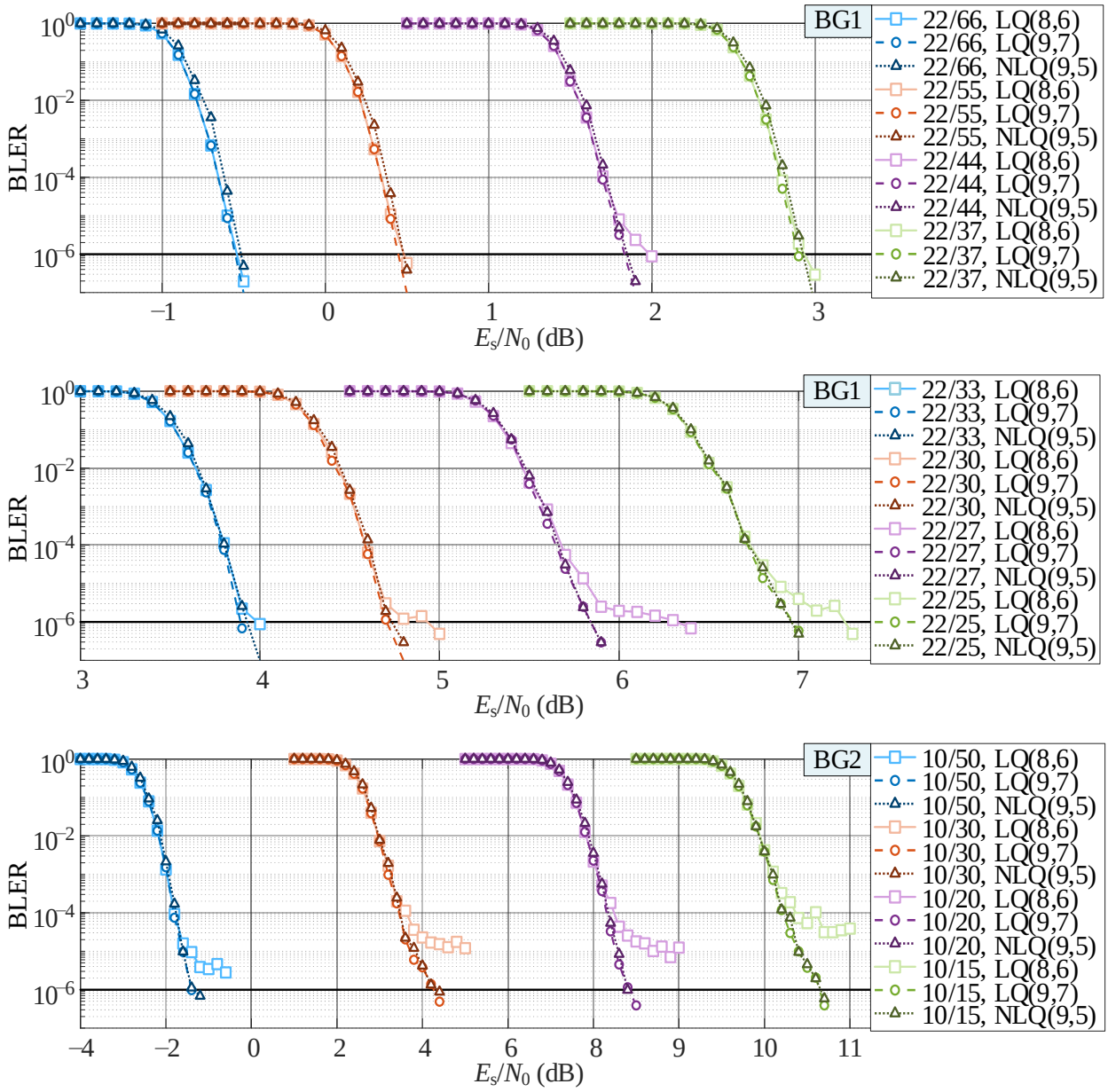
Слика 4.24. Архитектура декодера за хибридно декодовање са нелинеарном квантизацијом порука



Слика 4.25. Функција квантизатора за нелинеарну квантизацију апсолутних вредности порука варијабилних чворова и функција деквантизатора са применом офсета за поруке контролних чворова

На слици 4.26 је приказано поређење перформанси контроле грешака за предложени декодер са нелинеарном квантизацијом и претходно описане декодере са линеарном квантизацијом порука, за два различита формата порука и LLR вредности. Резултати

показују да се проширењем динамичког опсега порука на седам бита и LLR вредности на девет бита достижу вероватноће грешке по кодној речи мање од 10^{-6}. Међутим, исти закључак важи и за декодер код кога су поруке нелинеарно квантизоване на пет бита. Може се приметити мали губитак (< 0,05 dB) у односу на линеарну квантизацију за кодове изведене из основног графа 1 и то за мале кодне количнике.



Слика 4.26. Перформансе контроле грешака декодера за хибридно декодовање са линеарном квантизацијом порука на 6 бита и 8 бита за LLR вредности (LQ(8,6)), са линеарном квантизацијом порука на 7 бита и 9 бита за LLR вредности (LQ(9,7)) и са нелинеарном квантизацијом порука на 5 бита и 9 бита за LLR вредности (NLQ(9,5)). Величина циркуланта за све кодове је 384.

Коначно, резултати имплементације на Xilinx Zynq Ultrascale+ RF-SoC чипу (XCZU28DR) за све три поменуте шеме квантизације приказани су у табели 4.3. Очекивано, реализација која користи линеарну квантизацију и седам бита за поруке заузима највећу количину FPGA ресурса. Међутим, занимљиво је упоредити реализацију која користи нелинеарну квантизацију и реализацију која користи линеарну квантизацију и шест бита за поруке. Заузеће ресурса је приближно једнако. Декодер који користи нелинеарну квантизацију чак користи мање логичких елемената (лукап табела), што узрокује да је и

укупан број Slice блокова мањи. Ипак, број регистара и RAM блокова је већи него код декодера са линеарном квантизацијом.

Табела 4.3. Резултати имплементације 5G NR декодера за хибридно декодовање за различите шеме квантизације порука и LLR вредности

Квантизација	Заузеће ресурса				Потребна меморија у битима			$f_{\max}$ (MHz)
	Slice	LUT	FF	BRAM	LLR	$M_{c\to v}$	Укупно	
Линеарна (8, 6)	17667	96625	**87751**	**136,5**	**497664**	728064	1225728	404
Линеарна (9, 7)	21606	117148	108429	152	559872	849408	1409280	404
Нелинеарна (9, 5)	**17294**	**90626**	103746	141,5	559872	**606720**	**1166592**	**408**

Поред поменутих резултата, у табели 4.3 приказан је и укупан број бита потребних за смештање свих LLR вредности (за меморију за декодовање, бафер за улазне и излазне податке и бафер потребан за ажурирање код хибридног декодовања) и свих порука контролних чворова. Ови бројеви су од значаја у случају да је декодер потребно имплементирати на чипу посебне намене (енгл. *Application Specific Integrated Circuit, ASIC*). Тада се RAM блокови који су у FPGA реализацији непопуњени мењају меморијским блоковима минимално потребног капацитета. Број бита потребан за LLR вредности може се израчунати помоћу следећег израза:

$$M_{\mathrm{LLR}} = n_{\mathrm{b}} \cdot Z_{\max} \cdot b_{\mathrm{LLR}} + n_{\mathrm{b}} \cdot Z_{\max} \cdot b_{\mathrm{LLR}} + \left(k_{\mathrm{b}} + 4\right) \cdot Z_{\max} \cdot b_{\mathrm{LLR}} = \left(2 \cdot n_{\mathrm{b}} + k_{\mathrm{b}} + 4\right) \cdot 384 \cdot b_{\mathrm{LLR}}, \quad (102)$$

где је b_{LLR} битска ширина LLR вредности. Први сабирак у изразу (102) је број бита потребних за меморију за декодовање, други сабирак број бита за улазно-излазни бафер, а трећи сабирак број бита потребних за бафер за чување LLR вредности неопходних за ажурирање у конфликтним ситуацијама. Број бита потребан за поруке контролних чворова се може израчунати помоћу следећег израза:

$$M_{\mathrm{M}_{\mathrm{c}\to\mathrm{v}}} = Z_{\max} \cdot N_{\mathrm{circ,max}} \cdot b_{\mathrm{M}_{\mathrm{c}\to\mathrm{v}}} = 384 \cdot 316 \cdot b_{\mathrm{M}_{\mathrm{c}\to\mathrm{v}}}, \quad (103)$$

где је $N_{\mathrm{circ,max}}$ максимални број циркуланата у контролној матрици када се у обзир узимају сви подржани кодови, док је $b_{\mathrm{M}_{\mathrm{c}\to\mathrm{v}}}$ број бита за представу поруке. С обзиром на то да битска ширина порука више доприноси укупној потребној меморији у битима него битска ширина LLR вредности, нелинеарном квантизацијом се постиже и одређена уштеда.

На основу приказаних резултата може се закључити да се нелинеарном квантизацијом порука и проширењем динамичког опсега добија декодер за 5G NR стандард који има значајно боље перформансе контроле грешака уз упоредиво, или чак ефикасније, заузеће хардверских ресурса.

4.7. СМАЊЕЊЕ КОМПЛЕКСНОСТИ ДЕКОДОВАЊА 5G NR КОДОВА

Иако алгоритми засновани на Min-Sum апроксимацији омогућавају значајно смањење комплексности у поређењу са оригиналним SPA алгоритмом, могуће је извршити додатну редукцију комплексности ако се уместо израчунавања минимума и субминимума рачуна само минимум, а субминимум апроксимира [145]–[149]. Уштеде у хардверским ресурсима могу бити значајне посебно код реализација процесора контролног чвора код којих се више порука варијабилних чворова обрађује паралелно [146]–[149]. Стандардна серијска реализација јединице контролног чвора, код које су улазни и излазни део процесора

распрегнути једним степеном регистара, има мање користи од поменуте апроксимације, али је у раду [59] показано да се и за њу може остварити одређена уштеда у ресурсима. Међутим, ако се декодују нерегуларни кодови, какви су на пример кодови из 5G NR стандарда, за реализацију процесора контролних чворова потребан је FIFO бафер, који обезбеђује уклањање хазарда података услед превременог уписа, као што је показано у одељку 4.3.2. Ресурси које тај бафер заузима директно зависе од потребе за чувањем другог минимума, па би коришћење алгоритма код кога се тачно израчунава само један минимум могло допринети значајно смањеном заузећу хардверских ресурса.

Први алгоритам који користи минимум, а субминимум апроксимира сабирањем константног тежинског фактора и израчунатог минимума предложен је у [145]. Алгоритам је у тексту означен са smOMS (од енгл. *single minimum offset Min-Sum*). Иако најједноставнији, овај приступ има велике губитке у перформансама контроле грешака, а посебно се лоше понаша у региону ниске вероватноће грешке. Перформансе се могу побољшати ако тежински фактор није фиксан, већ се мења у различитим итерацијама декодовања, што је предложено у [146]. Наиме, примећено је да се разлике између субминимума и минимума код стандардних MS алгоритама повећавају у каснијим итерацијама декодовања, па је и тежински фактор потребно прогресивно повећавати како декодовање одмиче. У наставку, овај алгоритам је означен са vwsmOMS (од енгл. *variable weight smOMS*).

У овом потпоглављу приказан је процесор контролног чвора који у основи користи vwsmOMS алгоритам, али уз одређене измене које доприносе побољшању перформанси контроле грешака. Додатно, предложен је другачији метод подешавања тежинског фактора којим се перформансе могу додатно побољшати ако се декодују нерегуларни LDPC кодови.

4.7.1. Алгоритам и архитектура процесора контролног чвора

Контролни чвор vwsmOMS алгоритма генерише поруке на основу следећег израза:

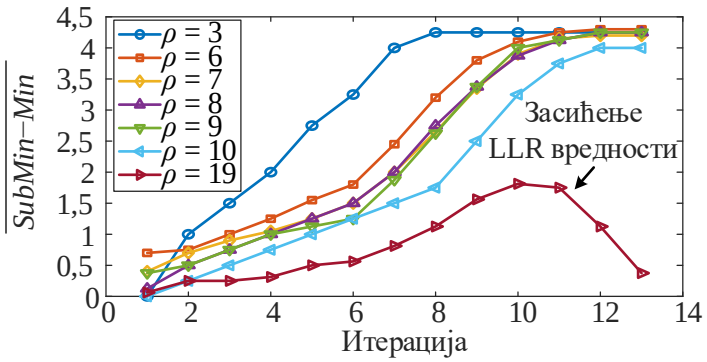
$$M_{c\to v}=\left(\prod_{v'\in V_c\setminus v}\operatorname{sgn}\left(M_{v'\to c}\right)\right)\cdot\max\left(M_{\mathrm{e}}-\beta,0\right), \tag{104}$$

при чему је

$$M_{\mathrm{e}}=\begin{cases}\min\limits_{v'\in V_c}\left(\left|M_{v'\to c}\right|\right)+w, & \text{ако је } \min\limits_{v'\in V_c}\left(\left|M_{v'\to c}\right|\right)=\left|M_{v\to c}\right|\wedge N_{\min}=1\\ \min\limits_{v'\in V_c}\left(\left|M_{v'\to c}\right|\right), & \text{иначе}\end{cases}, \tag{105}$$

где је w тежински фактор за естимацију субминимума, а $N_{\min}$ је број апсолутних вредности порука контролних чворова које су једнаке минимуму. Дакле, ако су минимум и субминимум једнаки, израчунавање порука је исто као код оригиналног OMS алгоритма. Вредности порука су мале у првим итерацијама декодовања, па се често дешава да постоји више од једне поруке која је једнака минимуму. Међутим, врло брзо, након неколико итерација декодовања, разлике између апсолутних вредности порука постају значајне. На слици 4.27 илустровано је ово понашање на примеру кода (15360, 8448) из 5G NR стандарда. За све контролне чворове једнаке тежине, приказана је зависност просечне вредности разлике субминимума и минимума од редног броја итерације при декодовању OMS алгоритмом.

За контролне чворове велике тежине може се приметити да просечна вредност разлике опада при крају декодовања, што се објашњава засићењем LLR вредности у реализацији алгоритма која користи представе бројева са фиксним зарезом. Тада велики број порука има велике, засићене апсолутне вредности, па је и разлика између минимума и субминимума једнака нули, што утиче на просечну вредност.



Слика 4.27. Зависност просечне вредности разлике субминимума и минимума од редног броја итерације при декодовању 5G NR кода (15360, 8448) Min-Sum алгоритмом са офсетом [59]

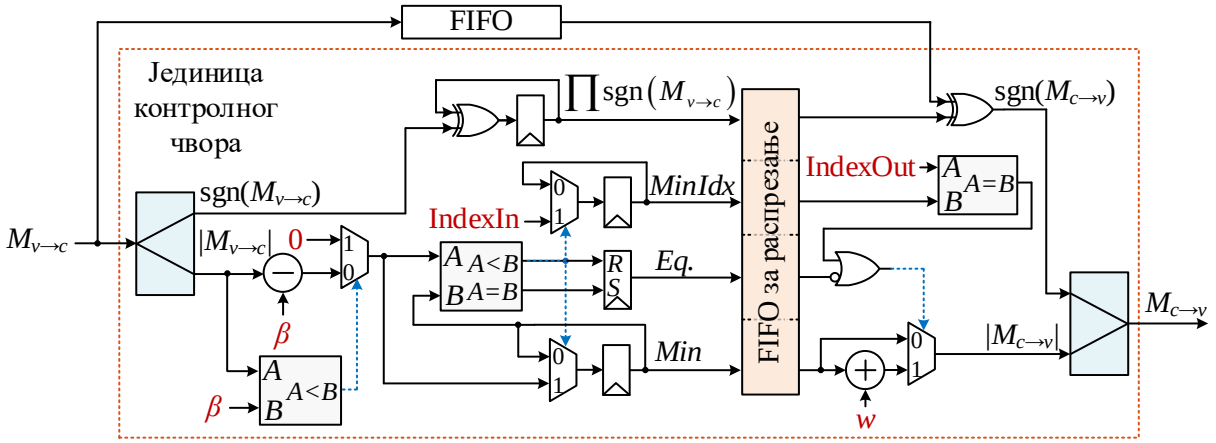
С обзиром на то да процесор контролног чвора vwsmOMS алгоритма ради исто као процесор OMS алгоритма у ситуацијама када је број апсолутних вредности порука које су једнаке минимуму већи од један, у овој дисертацији и у претходном раду [60] предложена је модификација редоследа израчунавања унутар јединице контролног чвора. Наиме, одузимање офсета не мора нужно бити на излазу. Исти резултати би се добили ако би се примена офсета код оригиналног алгоритма извршила на улазу, тј. над апсолутним вредностима порука варијабилних чворова. Међутим, када се користи само један минимум, одузимањем офсета на почетку постиже се да је већи број порука међу којима се тражи минимум једнак нули, па је чешћа ситуација да се алгоритам понаша као оригинални OMS алгоритам. Због тога се могу очекивати побољшане перформансе контроле грешака без повећања комплексности. Стога је функција која се имплементира у контролном чвору дефинисана следећим сетом једначина:

$$
\begin{aligned}
M_{\mathrm{e}} &= \min_{v'\in V_c}\left(\max\left(\left|M_{v'\to c}\right|-\beta,0\right)\right),\\
v_{\min} &= \arg\min_{v'\in V_c}\left(\max\left(\left|M_{v'\to c}\right|-\beta,0\right)\right),\\
\left|M_{c\to v}\right| &= \begin{cases} M_{\mathrm{e}}+w, & \text{ако је } v_{\min}=v\wedge N_{\min}=1\\ M_{\mathrm{e}}, & \text{иначе}\end{cases}\\
M_{c\to v} &= \left|M_{c\to v}\right|\cdot\left(\prod_{v'\in V_c\setminus v}\operatorname{sgn}\left(M_{v'\to c}\right)\right).
\end{aligned}
\tag{106}
$$

Архитектура јединице контролног чвора којом се реализује vwsmOMS алгоритам са горепоменутим измењеним редоследом израчунавања приказана је на слици 4.28. Одузимање офсета је померено одмах након конверзије представе поруке варијабилног чвора. Након тога се израчунава само један минимум и његов индекс, као и производ свих знакова. Поред ових вредности, алгоритам захтева и информацију о томе да ли је више од једне вредности једнако минималној. Она се добија у току серијског израчунавања минимума провером да ли је нека од примљених вредности једнака минималној, па се у тој ситуацији сетује флип-флоп чији је излаз сигнал *Eq*. Ако се на улазу појави нова минимална вредност, флип-флоп се ресетује.

Поред измењене архитектуре контролног чвора, како би се побољшале перформансе контроле грешака, могуће је одабрати другачији начин подешавања тежинског фактора него што је то предложено у оригиналном vwsmOMS алгоритму [146]. Посматрајући слику 4.27, може се закључити да су просечне разлике субминимума и минимума за чворове мање тежине веће од разлика за чворове веће тежине. Због тога је у овој дисертацији vwsmOMS модификован тако да параметар w расте са порастом редног броја итерације, али са

различитом функцијом за различите контролне чворове. Овакав алгоритам је скраћено назван EvwsmOMS (од енгл. *Enhanced vwsmOMS*).



Слика 4.28. Јединица контролног чвора за OMS алгоритам са једним минимумом са варијабилним тежинским фактором за естимацију другог минимума [60]

4.7.2. Перформансе контроле грешака и заузеће ресурса

Како би се испитао утицај смањења комплексности израчунавања у контролном чвору на перформансе контроле грешака, извршене су Монте Карло симулације за три кода из 5G NR стандарда кодних количника 22/55, 22/36 и 22/26. Сва три кода су изведена из основног графа 1 и величина циркуланта им је $Z = 384$. Симулације су извршене за AWGN канал и QPSK модулацију, при чему је максималан број итерација 20. Тестирани су Min-Sum алгоритам (MS), Min-Sum алгоритам са офсетом (OMS), OMS алгоритам који користи само један минимум, а други апроксимира додавањем константног фактора (smOMS), smOMS алгоритам са прогресивним повећавањем тежинског фактора (vwsmOMS) и предложени метод којим се тежински фактор подешава другачије за различите контролне чворове. Поред перформанси оригиналних верзија алгоритама, тестиране су и перформансе алгоритама код којих је примена офсета померена на улаз процесора контролног чвора. Ове верзије су обележене додатном ознаком **S** (од енгл. *swapped*). Параметри тестираних алгоритама приказани су у табели 4.4, а остварене перформансе контроле грешака на слици 4.29. За све симулације је коришћено оптимизовано хибридно декодовање.

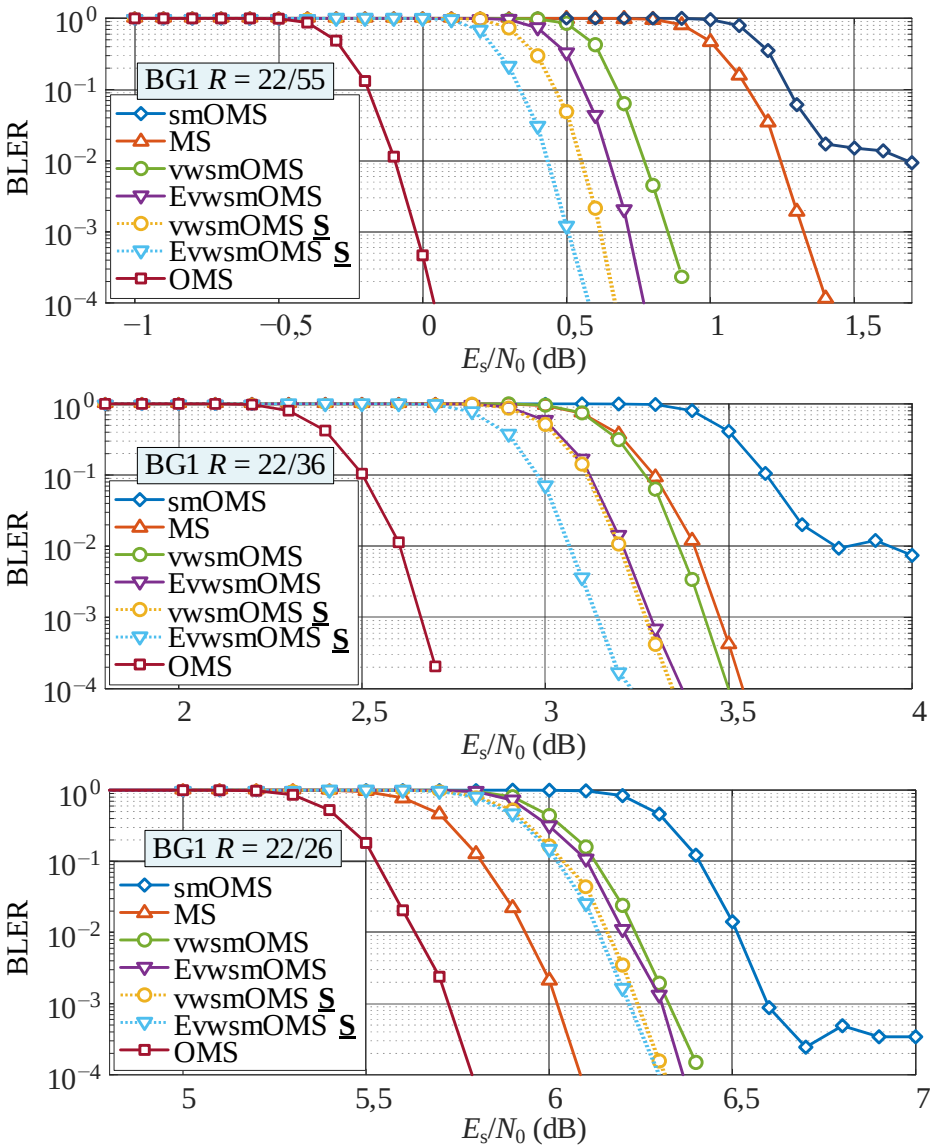
Табела 4.4. Вредности тежинског фактора коришћене у OMS алгоритмима са једним минимумом

	5G NR LDPC код					
	(21120, 8448); $R = 22/55$		(13824, 8448); $R = 22/36$		(9984, 8448); $R = 22/26$	
Алгоритам	ρ	w	ρ	w	ρ	w
smOMS	све вредности	2	све вредности	2,25	све вредности	3,5
vwsmOMS	све вредности	$0,22+0,22it$	све вредности	$0,22+0,22it$	све вредности	$0,23+0,23it$
EvwsmOMS (ова дисертација)	3	$0,5+0,4it$	3	$0,5+0,4it$	3	$0,5+0,4it$
	4, 5	$0,25+0,25it$	6, 7, 8, 9, 10	$0,25+0,25it$	8	$0,25+0,25it$
	6, 7, 8, 9, 10	$0,25+0,22it$				
	19	$0,24it$	19	$0,22+0,22it$	19	$0,22+0,22it$

Из добијених резултата се најпре може закључити да је smOMS практично неупотребљив за декодовање 5G NR кодова, с обзиром на то да за скоро све кодне количнике не достиже мале вероватноће грешке, док су перформансе осталих алгоритама значајно боље.

Перформансе Min-Sum алгоритма са офсетом су, очекивано, најбоље јер су остали алгоритми добијени његовим апроксимацијама. Губитак који у односу на њега постоји за Min-Sum алгоритам без офсета, веома је различит за различите кодне количнике. За мале вредности кодног количника, губитак је изузетно велики (преко 1 dB), док је за велике вредности кодног количника губитак значајно мањи (око 0,3 dB).

Поредећи перформансе алгоритама који користе један минимум, при чему се други минимум апроксимира сабирањем са променљивим тежинским параметром, може се закључити да измена редоследа израчунавања доприноси побољшању перформанси од око 0,1 dB, док модификован одабир тежинских параметара омогућава додатно побољшање од око 0,1 dB. Ипак, најбоља верзија овог алгоритма и даље показује значајан губитак у односу на оригинални OMS. У табели 4.5 приказано је поређење перформанси контроле грешака за MS, OMS и најбољи OMS који користи један минимум.



Слика 4.29. Перформансе различитих MS алгоритама за кодове из 5G NR стандарда [60]

Предложена архитектура јединице контролног чвора за vwsmOMS алгоритам, а уједно и EvwsmOMS, имплементирана је на Xilinx Zynq Ultrascale+ RF-SoC чипу (XCZU28DR). Резултати имплементације 384 паралелна процесора за поменути алгоритам и поређење са процесорима за MS и OMS приказани су у табели 4.5. Из резултата се види да се коришћењем EvwsmOMS алгоритма остварује значајна уштеда у односу на OMS. Међутим, треба рећи да је уштеда од преко 35% остварена само у јединицама контролних чворова. Остали блокови захтевају исте хардверске ресурсе као и раније, што чини да је уштеда у ресурсима за цео декодер свега око 5%. Имајући у виду губитке у перформансама контроле грешака, остварени резултат не оправдава коришћење EvwsmOMS алгоритма.

Табела 4.5. Резултати имплементације за 384 процесора контролног чвора различитих алгоритама заснованих на Min-Sum алгоритму и поређење перформанси контроле грешака

Алгоритам	Заузеће ресурса			E_s/N_0 (dB) за BLER $=10^{-4}$		
	Slice	LUT	FF	$R=22/55$	$R=22/36$	$R=22/26$
MS	1892	11227	6573	1,4	3,6	6,1
OMS	2065	12770	6573	**0,1**	**2,7**	**5,8**
EvwsmOMS **S**	**1513**	**8265**	**4935**	0,6	3,2	6,3

5. Закључак

У дисертацији су приказане нове хардверске архитектуре за кодовање и декодовање структурираних LDPC кодова. Поред архитектуралних решења, предложени су и алгоритамски доприноси који омогућавају убрзање рада и оптимизацију перформанси контроле грешака. Реализације кодера и декодера подржавају велики степен флексибилности, која је један од основних захтева у савременим комуникационим системима. Посебна пажња посвећена је ефикасности искоришћења хардверских ресурса изражене преко количника протока и количине употребљених ресурса. Оптимизација хардвера по овом критеријуму омогућава остваривање произвољно великог протока уз минимално заузеће ресурса. Докле год је кашњење једне компоненте, било кодера било декодера, прихватљиво за систем, за остварење великог протока прагматично је користити више хардверски ефикасних компоненти, које раде у паралели, пре него једну изузетно паралелизовану али неоптималну компоненту. Тиме се, поред смањене количине хардверских ресурса, постиже и боља енергетска ефикасност. Решења предложена у дисертацији у многоме доприносе побољшањима у том погледу.

У првом делу дисертације пројектован је кодер за 5G NR комуникациони стандард, а којим се могу кодовати сви LDPC кодови дефинисани спецификацијама, при чему су остварени велики информациони протоци не само за дугачке кодне речи, већ и за кодне речи средње и мале дужине. Архитектура кодера је заснована на серијском методу кодовања која обрађује један циркулант контролне матрице по циклусу сигнала такта. Међутим, за краће кодне речи омогућено је да се хардверски ресурси, који иначе нису у употреби, искористе за обраду већег броја циркуланата. Ово је пре свега постигнуто иновативним дизајном мреже за кружни померај која може да ротира једну секвенцу података ако је она велике дужине, што одговара дугачким кодним речима, али и више независних секвенци података када су оне краће дужине, што се дешава при кодовању краћих кодова. Ипак, при процесирању контролне матрице у оригиналном редоследу, често није могуће постићи да се обрађује више циркуланата у паралели. Због тога је редослед процесирања оптимизован коришћењем генетичког алгоритма тако да се оствари што је могуће краће време кодовања када постоје ресурси за обраду више циркуланата у једном такту. Оптимизација је омогућила велика побољшања у оствареном протоку (до 183%) и кашњењу кодера (до 92%). Кодер реализован на FPGA хардверској платформи остварује вишегигабитске протоке и бољу ефикасност искоришћења хардверских ресурса од решења присутних у литератури. Ефикасност предложеног кодера је и до пет пута већа од ефикасности изузетно паралелних архитектура, док је кашњење кодовања задржано на вредностима испод 1 μs.

Поред примене у 5G NR стандарду, показано је да се остварени доприноси могу искористити и за кодовање других квазицикличних LDPC кодова. Са циљем повећања протока, оптимизација редоследа процесирања заснована на генетичком алгоритму може се употребити у било којој архитектури која омогућава паралелно процесирање више врста пермутационе матрице кода. Додатно, предложени метод за пројектовање мреже за кружни

померај може се применити у било којој архитектури за коју је потребно подржати више режима ротације улазних података. Као пример приказан је кодер LDPC кодова из WiMAX комуникационог стандарда.

Други део дисертације обрађује алгоритам декодовања и архитектуру декодера за квазицикличне LDPC кодове и структуриране LDPC кодове са акумулацијом бита парности. Декодер представљен у дисертацији флексибилнији је од кодера, јер се њиме могу декодовати сви квазициклични кодови докле год је величина циркуланта подржана мрежом за кружни померај.

Алгоритам декодовања и архитектура декодера засновани су на методу слојевитог декодовања код кога је једна итерација декодовања подељена на подитерације, у оквиру којих се обрађују различити делови контролне матрице. Како би се повећала учестаност сигнала такта, а самим тим и проток декодера, неопходно је користити технику проточне обраде. Проточна обрада узрокује ситуације у којима је потребно прочитати стања варијабилних чворова за извршавање једне подитерације пре него што су она ажурирана, јер се претходне итерације нису завршиле. Због тога се морају увести циклуси паузе како би се избегли хазарди података услед превременог читања. Међутим, за декодер предложен у дисертацији не постоје циклуси паузе, па је број циклуса такта потребан за једну итерацију декодовања једнак броју циркуланата у контролној матрици кода, невезано од тога који је кôд примењен. Како би се елиминисали циклуси паузе, предложен је алгоритам хибридног декодовања, који функционише на исти начин као слојевито декодовање када нема хазарда података. Међутим, у ситуацијама када би код слојевитог декодовања требало увести циклусе паузе, декодовање се наставља са неажурним стањима варијабилних чворова, али се доприноси контролних чворова додају на другачији начин него код слојевитог декодовања. Суштински, декодер тада прелази на симултано декодовање које се одвија у субграфу кода.

С обзиром на то да се симултаним декодовањем добијају лошије перформансе контроле грешака од перформанси слојевитог декодовања, посебно при малом броју итерација, и код предложеног хибридног алгоритма декодовања могу се јавити одређени губици. Како би се они надокнадили, у дисертацији је представљен метод за оптимизацију редоследа процесирања контролне матрице којим се добијају најбоље перформансе, а предложено је да се преостали губици надокнаде додатним итерацијама. Узимајући у обзир чињеницу да је за хибридно декодовање потребно много мање циклуса такта за извршавање једне итерације декодовања, чак и након додавања допунских итерација, остварују се велика побољшања у протоку и кашњењу. На пример, ако се користе кодови из 5G NR стандарда и декодер са 13 степени проточне обраде, проток предложеног декодера је већи за између 30,8% и 109,1% од протока декодера за слојевито декодовање и то за исте перформансе контроле грешака.

Хибридним декодовањем омогућено је раскидање везе између два опречна захтева: повећања броја степени проточне обраде, ради повећања учестаности сигнала такта, и смањења броја степени проточне обраде, ради смањења броја циклуса паузе при слојевитом декодовању. На тај начин је омогућена произвољна дубина проточне обраде и висока учестаност сигнала такта, а самим тим и велики проток и ефикасност искоришћења хардверских ресурса. Као примери, декодери имплементирани за WiMAX, DVB-S2x и 5G NR стандарде остварују вишегигабитске протоке и значајно унапређење у ефикасности када се упореде са референтним резултатима из литературе.

Поред хазарда података који настају услед превременог читања стања варијабилних чворова, при декодовању нерегуларних кодова могући су и хазарди услед превременог уписа међурезултата унутар процесора контролних чворова. Они се такође морају избећи увођењем циклуса паузе чији број зависи од тежина контролних чворова из слојева који се сукцесивно обрађују. Што је већа разлика између тежина чворова суседних слојева, то је већи број циклуса паузе који се мора увести. У дисертацији је предложено коришћење FIFO бафера унутар процесора контролних чворова који служе за распрезање два зависна блока за израчунавање. Ова измена доприноси одређеном повећању заузећа хардверских ресурса, али

у потпуности елиминише поменуте циклусе паузе. Без коришћења FIFO бафера, не би било могуће да се паузе избегну, па би оптимизација редоследа процесирања морала да буде вишекритеријумска. Било би неопходно здружено оптимизовати перформансе контроле грешака и трајање декодовања, што је значајно захтевнији задатак. Раније је поменуто да нерегуларни кодови у општем случају имају боље перформансе контроле грешака од регуларних, а посебно су значајни у каналима у којима не делује само бели Гаусов шум [141]. У том смислу, предложена архитектура има још већи значај, јер омогућава декодовање нерегуларних кодова произвољног профила без иједног циклуса паузе. Стога је интересантан приступ здруженог дизајна кода и декодера, који се може применити у даљим истраживањима. С обзиром на то да декодер из дисертације нема ограничења у погледу профила кода, дизајнеру кодова се може дати велика слобода за избор различитих профила. Наравно, кôд који се пројектује мора бити квазицикличан или налик на структуриране кодове из DVB-S2(x) стандарда. Ипак, велика дубина проточне обраде утиче на број конфликтних ситуација и у одређеној мери, без обзира на оптимизацију, на перформансе контроле грешака, па је занимљиво питање какав је кôд оптималан за дати декодер.

Поред горепоменутих доприноса, извршена је детаљнија анализа перформанси контроле грешака за предложену хардверску реализацију декодера. Примећено је да се услед ограниченог динамичког опсега, тј. ограниченог броја бита за представу порука и стања варијабилних чворова, јавља ефекат заравњења карактеристике вероватноће грешке (BLER криве). У дисертацији је предложено решење у виду нелинеарне квантизације порука и делимично измењена архитектура декодера, којом се постижу значајно побољшане перформансе без већих захтева за додатним хардверским ресурсима. Даља истраживања би могла ићи и ка даљој оптимизацији битских ширина и представа бројева којом се остварују уштеде у ресурсима, додатно побољшане перформансе контроле грешака или и једно и друго.

У дисертацији је приказано и једно решење за смањење комплексности декодовања 5G NR кодова, коришћењем архитектуре контролног чвора која користи само један минимум, али је закључено да остварене уштеде у ресурсима не оправдавају значајан губитак у перформансама контроле грешака. Ипак, представљене идеје могу бити од значаја ако се декодују други кодови. У том смислу, истраживање из дисертације се може наставити у смеру проналаска алгоритма декодовања који има смањену комплексност, а нема велики губитак у перформансама или који за сличну комплексност остварује добитак у перформансама контроле грешака.

У архитектуралном смислу, свакако је значајан даљи рад на архитектурама мрежа за кружни померај, с обзиром на то да савремени комуникациони стандарди захтевају све већу флексибилност, а флексибилне мреже за кружни померај заузимају велики проценат хардверских ресурса. Поред тога, мрежа за кружни померај која има више режима рада може се применити и у декодеру тако да се за кодне речи средње и мале дужине остваре већи протоци и мања кашњења, тј. може се применити иста идеја која је коришћена за дизајн кодера. Тада би било потребно урадити додатне оптимизације редоследа процесирања које омогућавају да се већи број слојева ефикасно обрађује у паралели, а да се истовремено смањи број конфликтних ситуација које настају због проточне обраде. За потребе ажурирања резултата паралелно процесираних слојева, може се користити ажурирање слично ажурирању примењеном у конфликтним ситуацијама код хибридног декодовања. Таква архитектура би потенцијално имала деградације у перформансама контроле грешака, јер је ближа архитектури за симултано декодовање, али би се могла постићи велика убрзања.

На крају, с обзиром на то да је предложена архитектура декодера флексибилна и да дизајн контролне јединице која управља радом целог система не зависи од примењеног кода, могло би се реализовати софтверско окружење које генерише код за опис хардвера, HDL кôд (енгл. *Hardware Description Language, HDL*), којим се описује декодер за произвољан квазицикличан или структурирани IRA код. Главни изазов у реализацији те идеје је

генератор мреже за кружни померај. Један пример за другачију архитектуру декодера приказан је у [40]. Софтверски генератор би требало да подржи унос кључних параметара кода (дужину кодне речи, величину циркуланта, максималан број циркуланата у контролној матрици, максималне и минималне тежине чворова, алгоритам декодовања примењен у јединици контролног чвора, битске ширине за представу различитих вредности и сл.) и да на основу њих генерише HDL код, који се може синтетисати за FPGA или ASIC чип. Вредност оваквог генератора је вишеструка. Поред очигледног убрзања процеса пројектовања декодера, таквим генератором дизајнеру кода се омогућава да веома брзо тестира перформансе контроле грешака за различите алгоритме декодовања и то за изузетно мале вредности вероватноће грешке.

Литература

[1] C. E. Shannon, “A mathematical theory of communication,” *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, Jul. 1948, doi: 10.1002/j.1538-7305.1948.tb01338.x.

[2] C. E. Shannon, “Communication in the presence of noise,” *Proc. IRE*, vol. 37, no. 1, pp. 10–21, Jan. 1949, doi: 10.1109/JRPROC.1949.232969.

[3] C. E. Shannon, “The zero error capacity of a noisy channel,” *IEEE Trans. Inform. Theory*, vol. 2, no. 3, pp. 8–19, Sep. 1956, doi: 10.1109/TIT.1956.1056798.

[4] C. E. Shannon, “Certain results in coding theory for noisy channels,” *Information and Control*, vol. 1, no. 1, pp. 6–25, Sep. 1957, doi: 10.1016/S0019-9958(57)90039-6.

[5] C. E. Shannon, “Probability of error for optimal codes in a Gaussian channel,” *Bell System Technical Journal*, vol. 38, no. 3, pp. 611–656, May 1959, doi: 10.1002/j.1538-7305.1959.tb03905.x.

[6] C. E. Shannon, “Coding theorems for a discrete source with a fidelity criterion,” vol. 7, pp. 142–163, 1959, doi: 10.1109/9780470544242.ch21.

[7] Д. Драјић и П. Иваниш, *Увод у теорију информација и кодовање*. Београд: Академска мисао, 2018.

[8] N. Wiener, “Generalized harmonic analysis,” *Acta Math.*, vol. 55, pp. 117–258, 1930, doi: 10.1007/BF02546511.

[9] N. Wiener, *The extrapolation, interpolation, and smoothing of stationary time series with engineering applications*. New York: John Wiley & Sons, Inc., 1949.

[10] Д. Драјић, *Увод у статистичку теорију телекомуникација*. Београд: Академска мисао, 2006.

[11] R. M. Fano, *Transmission of information: a statistical theory of communications*. New York: John Wiley & Sons, Inc., 1961.

[12] D. Huffman, “A method for the construction of minimum-redundancy codes,” *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, Sep. 1952, doi: 10.1109/JRPROC.1952.273898.

[13] N. Faller, “An adaptive system for data compression,” in *Record of the 7th Asilomar Conference on Circuits, Systems and Computers*, 1973, pp. 593–597.

[14] R. Gallager, “Variations on a theme by Huffman,” *IEEE Trans. Inform. Theory*, vol. 24, no. 6, pp. 668–674, Nov. 1978, doi: 10.1109/TIT.1978.1055959.

[15] J. Ziv and A. Lempel, “A universal algorithm for sequential data compression,” *IEEE Trans. Inform. Theory*, vol. 23, no. 3, pp. 337–343, May 1977, doi: 10.1109/TIT.1977.1055714.

[16] C. E. Shannon, “Communication theory of secrecy systems,” *Bell System Technical Journal*, vol. 28, no. 4, pp. 656–715, Oct. 1949, doi: 10.1002/j.1538-7305.1949.tb00928.x.

[17] 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*, Geneva, Switzerland, May 1993, vol. 2, pp. 1064–1070. doi: 10.1109/ICC.1993.397441.

[18] T. J. Richardson and R. L. Urbanke, “The capacity of low-density parity-check codes under message-passing decoding,” *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001, doi: 10.1109/18.910577.

[19] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, “Design of capacity-approaching irregular low-density parity-check codes,” *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001, doi: 10.1109/18.910578.

[20] Sae-Young Chung, G. D. Forney, T. J. Richardson, and R. Urbanke, “On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit,” *IEEE Commun. Lett.*, vol. 5, no. 2, pp. 58–60, Feb. 2001, doi: 10.1109/4234.905935.

[21] R. Gallager, “Low-density parity-check codes,” *IRE Trans. Inform. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962, doi: 10.1109/TIT.1962.1057683.

[22] D. J. C. MacKay and R. M. Neal, “Near Shannon limit performance of low density parity check codes,” *Electron. Lett.*, vol. 32, no. 18, pp. 1645–1646, Aug. 1996, doi: 10.1049/el:19961141.

[23] J. Pearl, *Probabilistic reasoning in intelligent systems*. San Mateo, CA, USA: Morgan Kaufman, 1988.

[24] F. R. Kschischang and B. J. Frey, “Iterative decoding of compound codes by probability propagation in graphical models,” *IEEE J. Select. Areas Commun.*, vol. 16, no. 2, pp. 219–230, Feb. 1998, doi: 10.1109/49.661110.

[25] K.-C. Ho, C.-L. Chen, and H.-C. Chang, “A 520k (18900, 17010) array dispersion LDPC decoder architectures for NAND flash memory,” *IEEE Trans. VLSI Syst.*, vol. 24, no. 4, pp. 1293–1304, Apr. 2016, doi: 10.1109/TVLSI.2015.2464092.

[26] IEEE Standard 802.3an™-2006, *IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—LAN/MAN—Specific Requirements Part 3: CSMA/CD Access Method and Physical Layer Specifications—Amendment: Physical Layer and Management Parameters for 10 Gb/s Operation, Type 10GBASE-T*. 2006.

[27] IEEE Standard 802.11™-2016, *IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. 2016.

[28] IEEE Standard 802.16™-2004, *IEEE Standard for Local and Metropolitan Area Networks Part 16: Air Interface for Fixed Broadband Wireless Access Systems*. 2004.

[29] Standard ETSI EN, 302 307–2 V.1.1.1 (2014-10), *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 2: DVB-S2 Extensions (DVB-S2X)*. 2014.

[30] Standard CM-SP-PHYv3.1-I11-170510, *DOCSIS 3.1: Data-Over-Cable Service Interface Specifications DOCSIS 3.1, Physical Layer Specification*. 2017.

[31] 3GPP TS 38.212 V16.5.0 (2021-03), *Technical Specification Group Radio Access Network; NR; Multiplexing and Channel Coding (Release 16)*. 2021.

[32] T. Richardson and S. Kudekar, "Design of low-density parity check codes for 5G new radio," *IEEE Commun. Mag.*, vol. 56, no. 3, pp. 28–34, Mar. 2018, doi: 10.1109/MCOM.2018.1700839.

[33] G. P. Fettweis, "The tactile internet: applications and challenges," *IEEE Veh. Technol. Mag.*, vol. 9, no. 1, pp. 64–70, Mar. 2014, doi: 10.1109/MVT.2013.2295069.

[34] S. K. Sharma, I. Woungang, A. Anpalagan, and S. Chatzinotas, "Toward tactile internet in beyond 5G era: recent advances, current issues, and future directions," *IEEE Access*, vol. 8, pp. 56948–56991, Mar. 2020, doi: 10.1109/ACCESS.2020.2980369.

[35] A. J. Blanksby and C. J. Howland, "A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder," *IEEE J. Solid-State Circuits*, vol. 37, no. 3, pp. 404–412, Mar. 2002, doi: 10.1109/4.987093.

[36] Z. Wang and Z. Cui, "Low-complexity high-speed decoder design for quasi-cyclic LDPC codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 1, pp. 104–114, Jan. 2007, doi: 10.1109/TVLSI.2007.891098.

[37] R. G. Gallager, "Analysis of number of independent decoding iterations," in *Low-density parity-check codes*, Cambridge, MA, USA: MIT Press, 1963, pp. 81–88.

[38] R. M. Tanner, D. Sridhara, A. Sridharan, T. E. Fuja, and D. J. Costello, "LDPC block and convolutional codes based on circulant matrices," *IEEE Trans. Inform. Theory*, vol. 50, no. 12, pp. 2966–2984, Dec. 2004, doi: 10.1109/TIT.2004.838370.

[39] M. P. C. Fossorier, "Quasi-cyclic low-density parity-check codes from circulant permutation matrices," *IEEE Trans. Inform. Theory*, vol. 50, no. 8, pp. 1788–1793, Aug. 2004, doi: 10.1109/TIT.2004.831841.

[40] P. Hailes, L. Xu, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, "A flexible FPGA-based quasi-cyclic LDPC decoder," *IEEE Access*, vol. 5, pp. 20965–20984, Mar. 2017, doi: 10.1109/ACCESS.2017.2678103.

[41] M. P. C. Fossorier, M. Mihaljević, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. Commun.*, vol. 47, no. 5, pp. 673–680, May 1999, doi: 10.1109/26.768759.

[42] J. Chen and M. P. C. Fossorier, "Density evolution for two improved BP-Based decoding algorithms of LDPC codes," *IEEE Commun. Lett.*, vol. 6, no. 5, pp. 208–210, May 2002, doi: 10.1109/4234.1001666.

[43] D. E. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *Proceedings on IEEE Workshop on Signal Processing Systems, 2004. SIPS 2004.*, Austin, Texas, USA, Oct. 2004, pp. 107–112. doi: 10.1109/SIPS.2004.1363033.

[44] R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. 27, no. 5, pp. 533–547, Sep. 1981, doi: 10.1109/TIT.1981.1056404.

[45] Creonic GmbH, "DVB-S2X transmitter and receiver product brief." 2019.

[46] Xilinx Inc., "LDPC encoder/decoder v2.0: LogiCORE IP product guide." Apr. 2018.

[47] D. A. Patterson and J. L. Hennessy, "An overview of pipelining," in *Computer organization and design: the hardware/software interface*, 4th ed., Burlington, MA: Morgan Kaufmann Publishers, 2009, pp. 330–344.

[48] C. Marchand, J.-B. Dore, L. Conde-Canencia, and E. Boutillon, “Conflict resolution for pipelined layered LDPC decoders,” in *Proceedings on 2009 IEEE Workshop on Signal Processing Systems*, Tampere, Finland, Oct. 2009, pp. 220–225. doi: 10.1109/SIPS.2009.5336255.

[49] V. L. Petrović, M. M. Marković, D. M. El Mezeni, L. V. Saranovac, and A. Radošević, “Flexible high throughput QC-LDPC decoder with perfect pipeline conflicts resolution and efficient hardware utilization,” *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 67, no. 12, pp. 5454–5467, Dec. 2020, doi: 10.1109/TCSI.2020.3018048.

[50] M. Rovini, G. Gentile, F. Rossi, and L. Fanucci, “A scalable decoder architecture for IEEE 802.11n LDPC codes,” in *Proceedings on 2007 IEEE Global Telecommunications Conference*, Washington, DC, USA, Nov. 2007, pp. 3270–3274. doi: 10.1109/GLOCOM.2007.620.

[51] X. Zhao, Z. Chen, X. Peng, D. Zhou, and S. Goto, “DVB-T2 LDPC decoder with perfect conflict resolution,” *IPSJ Trans. Syst. LSI Design Methodol.*, vol. 5, pp. 23–31, Feb. 2012, doi: 10.2197/ipsjtsldm.5.23.

[52] Z. Wu, D. Liu, and Y. Zhang, “Matrix reordering techniques for memory conflict reduction for pipelined QC-LDPC decoder,” in *Proceedings on 2014 IEEE/CIC International Conference on Communications in China (ICCC)*, Shanghai, China, Oct. 2014, pp. 354–359. doi: 10.1109/ICCChina.2014.7008301.

[53] C.-W. Sham, X. Chen, W. M. Tam, Y. Zhao, and F. C. M. Lau, “A layered QC-LDPC decoder architecture for high speed communication system,” in *Proceedings on 2012 IEEE Asia Pacific Conference on Circuits and Systems*, Kaohsiung, Taiwan, Dec. 2012, pp. 475–478. doi: 10.1109/APCCAS.2012.6419075.

[54] S. Kumawat, R. Shrestha, N. Daga, and R. Paily, “High-throughput LDPC-decoder architecture using efficient comparison techniques & dynamic multi-frame processing schedule,” *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 62, no. 5, pp. 1421–1430, May 2015, doi: 10.1109/TCSI.2015.2403032.

[55] “Implementation and performance of LDPC decoder,” 3GPP, Ericsson, Spokane, WA, USA, R1-1700111, Jan. 2017.

[56] O. Boncalo, G. Kolumban-Antal, A. Amaricai, V. Savin, and D. Declercq, “Layered LDPC decoders with efficient memory access scheduling and mapping and built-in support for pipeline hazards mitigation,” *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 66, no. 4, pp. 1643–1656, Apr. 2019, doi: 10.1109/TCSI.2018.2884252.

[57] P. Larrañaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, “Genetic algorithms for the travelling salesman problem: a review of representations and operators,” *Artif. Intell. Rev.*, vol. 13, no. 2, pp. 129–170, 1999, doi: 10.1023/A:1006529012972.

[58] V. L. Petrović, D. M. El Mezeni, and A. Radošević, “Flexible 5G new radio LDPC encoder optimized for high hardware usage efficiency,” *Electronics*, vol. 10, no. 9, p. 1106, May 2021, doi: 10.3390/electronics10091106.

[59] V. L. Petrović and D. M. El Mezeni, “Reduced-complexity offset min-sum based layered decoding for 5G LDPC codes,” in *Proceedings on 2020 28th Telecommunications Forum (TELFOR)*, Belgrade, Serbia, Nov. 2020, pp. 109–112. doi: 10.1109/TELFOR51502.2020.9306590.

[60] V. L. Petrović and D. M. El Mezeni, “Reduced-complexity offset Min-Sum check node unit for layered 5G LDPC decoder,” *Telfor Journal*, vol. 13, no. 1, pp. 7–12, Jul. 2021, doi: 10.5937/telfor2101007P.

[61] Siddiqi, Yu, and Joung, “5G Ultra-Reliable Low-Latency Communication implementation challenges and operational issues with IoT devices,” *Electronics*, vol. 8, no. 9, p. 981, Sep. 2019, doi: 10.3390/electronics8090981.

[62] M. C. Davey and D. J. C. MacKay, “Low density parity check codes over GF(q),” in *1998 Information Theory Workshop*, Killarney, Ireland, Jun. 1998, pp. 70–71. doi: 10.1109/ITW.1998.706440.

[63] Y. Toriyama and D. Markovic, “A 2.267-Gb/s, 93.7-pJ/bit non-binary LDPC decoder with logarithmic quantization and dual-decoding algorithm scheme for storage applications,” *IEEE J. Solid-State Circuits*, vol. 53, no. 8, pp. 2378–2388, Aug. 2018, doi: 10.1109/JSSC.2018.2832851.

[64] R. Klaimi, “Study of non-binary turbo codes for future communication and broadcasting systems,” Ecole nationale supérieure Mines-Télécom Atlantique, 2019.

[65] С. Бркић, “Декодовање кодова са малом густином провера парности у присуству грешака у логичким колима,” Универзитет у Београду - Електротехничку факултет, Београд, 2016.

[66] V. Mannoni, D. Declercq, and G. Gelle, “Optimized irregular low-density parity-check codes for multicarrier modulations over frequency-selective channels,” *EURASIP J. Adv. Signal Process.*, vol. 2004, no. 10, pp. 1546–1556, Sep. 2004.

[67] R. Comroe and D. Costello, “ARQ schemes for data transmission in mobile radio systems,” *IEEE J. Select. Areas Commun.*, vol. 2, no. 4, pp. 472–481, Jul. 1984, doi: 10.1109/JSAC.1984.1146084.

[68] D. J. C. MacKay, “Good error-correcting codes based on very sparse matrices,” *IEEE Trans. Inform. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999, doi: 10.1109/18.748992.

[69] P. Hailes, “Design and implementation of flexible FPGA-based LDPC decoders,” University of Southampton, Southampton, UK, 2018.

[70] D. J. C. MacKay and R. M. Neal, “Good codes based on very sparse matrices,” in *IMA International Conference on Cryptography and Coding, Lecture Notes in Computer Science*, Berlin, Germany, 1995, vol. 1025, pp. 100–111.

[71] Y. Kou, S. Lin, and M. P. C. Fossorier, “Low-density parity-check codes based on finite geometries: a rediscovery and new results,” *IEEE Trans. Inform. Theory*, vol. 47, no. 7, pp. 2711–2736, Nov. 2001, doi: 10.1109/18.959255.

[72] L. M. Batten, *Combinatorics of finite geometries*, 2nd ed. Cambridge ; New York: Cambridge University Press, 1997.

[73] R. M. Tanner, “A [155, 64, 20] sparse graph (LDPC) code,” presented at the IEEE International Symposium on Information Theory, Sorrento, Italy, Jun. 2000.

[74] T. Richardson and R. Urbanke, “Multi-edge type LDPC codes,” in *Workshop honoring Prof. Bob McEliece on his 60th birthday*, California Institute of Technology, Pasadena, California, May 2002, pp. 24–25.

[75] D. Divsalar, S. Dolinar, C. Jones, and K. Andrews, “Capacity-approaching protograph codes,” *IEEE J. Select. Areas Commun.*, vol. 27, no. 6, pp. 876–888, Aug. 2009, doi: 10.1109/JSAC.2009.090806.

[76] H. Li, B. Bai, X. Mu, J. Zhang, and H. Xu, “Algebra-assisted construction of quasi-cyclic LDPC codes for 5G new radio,” *IEEE Access*, vol. 6, pp. 50229–50244, Sep. 2018, doi: 10.1109/ACCESS.2018.2868963.

[77] D. Divsalar, H. Jin, and R. J. McEliece, “Coding theorems for turbo-like codes,” in *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, Allerton, IL, USA, 1998, vol. 36.

[78] H. Jin, A. Khandekar, and R. McEliece, “Irregular repeat-accumulate codes,” in *Proceedings on the 2nd International Symposium on Turbo Codes and Related Topics*, Brest, France, Sep. 2000, pp. 1–8.

[79] Xiao-Yu Hu, E. Eleftheriou, and D. M. Arnold, “Regular and irregular progressive edge-growth Tanner graphs,” *IEEE Trans. Inform. Theory*, vol. 51, no. 1, pp. 386–398, Jan. 2005, doi: 10.1109/TIT.2004.839541.

[80] D. V. Nguyen, S. K. Chilappagari, M. W. Marcellin, and B. Vasic, “On the construction of structured LDPC codes free of small trapping sets,” *IEEE Trans. Inform. Theory*, vol. 58, no. 4, pp. 2280–2302, Apr. 2012, doi: 10.1109/TIT.2011.2173733.

[81] C. Marchand, “Implementation of an LDPC decoder for the DVB-S2, -T2 and -C2 standards,” Université de Bretagne Sud, Lorient, France, 2010.

[82] S. Muller, M. Schreger, M. Kabutz, M. Alles, F. Kienle, and N. Wehn, “A novel LDPC decoder for DVB-S2 IP,” in *Proceedings on 2009 Design, Automation & Test in Europe Conference & Exhibition*, Nice, Apr. 2009, pp. 1308–1313. doi: 10.1109/DATE.2009.5090867.

[83] H. Wu and H. Wang, “A high throughput implementation of QC-LDPC codes for 5G NR,” *IEEE Access*, vol. 7, pp. 185373–185384, Dec. 2019, doi: 10.1109/ACCESS.2019.2960839.

[84] K. Andrews, S. Dolinar, and J. Thorpe, “Encoders for block-circulant LDPC codes,” in *Proceedings. International Symposium on Information Theory, 2005. ISIT 2005.*, Adelaide, Australia, 2005, pp. 2300–2304. doi: 10.1109/ISIT.2005.1523758.

[85] Zongwang Li, Lei Chen, Lingqi Zeng, S. Lin, and W. H. Fong, “Efficient encoding of quasi-cyclic low-density parity-check codes,” *IEEE Trans. Commun.*, vol. 54, no. 1, pp. 71–81, Jan. 2006, doi: 10.1109/TCOMM.2005.861667.

[86] H. Yasotharan and A. C. Carusone, “A flexible hardware encoder for systematic low-density parity-check codes,” in *Proceedings on 2009 52nd IEEE International Midwest Symposium on Circuits and Systems*, Cancun, Mexico, Aug. 2009, pp. 54–57. doi: 10.1109/MWSCAS.2009.5236155.

[87] D. Theodoropoulos, N. Kranitis, and A. Paschalis, “An efficient LDPC encoder architecture for space applications,” in *Proceedings on 2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS)*, Sant Feliu de Guixols, Spain, Jul. 2016, pp. 149–154. doi: 10.1109/IOLTS.2016.7604689.

[88] D. Chen, P. Chen, and Y. Fang, “Low-complexity high-performance low-density parity-check encoder design for China digital radio standard,” *IEEE Access*, vol. 5, pp. 20880–20886, Jul. 2017, doi: 10.1109/ACCESS.2017.2723046.

[89] A. Mahdi and V. Paliouras, “Simplified Multi-Level Quasi-Cyclic LDPC Codes for Low-Complexity Encoders,” in *Proceedings on 2012 IEEE Workshop on Signal Processing Systems*, Quebec City, QC, Oct. 2012, pp. 1–6. doi: 10.1109/SiPS.2012.21.

[90] A. Mahdi and V. Paliouras, “A low complexity-high throughput QC-LDPC encoder,” *IEEE Trans. Signal Process.*, vol. 62, no. 10, pp. 2696–2708, May 2014, doi: 10.1109/TSP.2014.2314435.

[91] A. Mahdi, N. Kanistras, and V. Paliouras, “A multirate fully parallel LDPC encoder for the IEEE 802.11n/ac/ax QC-LDPC codes based on reduced complexity XOR trees,” *IEEE Trans.*

Very Large Scale Integr. (VLSI) Syst., vol. 29, no. 1, pp. 51–64, Jan. 2021, doi: 10.1109/TVLSI.2020.3034046.

[92] T. J. Richardson and R. L. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 638–656, Feb. 2001, doi: 10.1109/18.910579.

[93] E. Eleftheriou and S. Olcer, "Low-density parity-check codes for digital subscriber lines," in *Conference proceedings on 2002 IEEE International Conference on Communications*, New York, NY, USA, May 2002, pp. 1752–1757. doi: 10.1109/ICC.2002.997149.

[94] T. Zhang and K. K. Parhi, "Joint (3, k)-regular LDPC code and decoder/encoder design," *IEEE Trans. Signal Process.*, vol. 52, no. 4, pp. 1065–1079, Apr. 2004, doi: 10.1109/TSP.2004.823508.

[95] H. Zhong and T. Zhang, "Block-LDPC: a practical LDPC coding system design approach," *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 52, no. 4, pp. 766–775, Apr. 2005, doi: 10.1109/TCSI.2005.844113.

[96] H. Zhang, J. Zhu, H. Shi, and D. Wang, "Layered approx-regular LDPC: code construction and encoder/decoder design," *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 55, no. 2, pp. 572–585, Mar. 2008, doi: 10.1109/TCSI.2008.916433.

[97] J. K. Kim, H. Yoo, and M. H. Lee, "Efficient encoding architecture for IEEE 802.16e LDPC codes," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. E91-A, no. 12, pp. 3607–3611, Dec. 2008, doi: 10.1093/ietfec/e91-a.12.3607.

[98] P. Zhang, S. Du, C. Liu, and Q. Jiang, "Fast encoding of quasi-cyclic low-density parity-check codes in IEEE 802.15.3c," *Electron. lett.*, vol. 51, no. 21, pp. 1713–1715, Oct. 2015, doi: 10.1049/el.2015.1770.

[99] X. Wang, T. Ge, J. Li, C. Su, and F. Hong, "Efficient multi-rate encoder of QC-LDPC codes based on FPGA for WIMAX standard," *Chin. J. Electron.*, vol. 26, no. 2, pp. 250–255, Mar. 2017, doi: 10.1049/cje.2017.01.006.

[100] A. E. Cohen and K. K. Parhi, "A low-complexity hybrid LDPC code encoder for IEEE 802.3an (10GBase-T) ethernet," *IEEE Trans. Signal Process.*, vol. 57, no. 10, pp. 4085–4094, Oct. 2009, doi: 10.1109/TSP.2009.2022919.

[101] D. Theodoropoulos, N. Kranitis, A. Tsigkanos, and A. Paschalis, "Efficient architectures for multigigabit CCSDS LDPC encoders," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 5, pp. 1118–1127, May 2020, doi: 10.1109/TVLSI.2020.2975050.

[102] Y. Sun, M. Karkooti, and J. Cavallaro, "High throughput, parallel, scalable LDPC encoder/decoder architecture for OFDM systems," in *Proceedings in 2006 IEEE Dallas/CAS Workshop on Design, Applications, Integration and Software*, Univ. of Texas at Dallas, Richardson, TX, USA, Oct. 2006, pp. 39–42. doi: 10.1109/DCAS.2006.321028.

[103] Z. Cai, J. Hao, P. H. Tan, S. Sun, and P. S. Chin, "Efficient encoding of IEEE 802.11n LDPC codes," *Electron. Lett.*, vol. 42, no. 25, pp. 1471–1472, Oct. 2006, doi: 10.1049/el:20063126.

[104] J. M. Pérez and V. Fernández, "Low-cost encoding of IEEE 802.11n," *Electron. Lett.*, vol. 44, no. 4, pp. 307–308, Feb. 2008, doi: 10.1049/el:20083140.

[105] Y.-M. Jung, C.-H. Chung, Y.-H. Jung, and J.-S. Kim, "7.7 Gbps encoder design for IEEE 802.11ac QC-LDPC codes," *J. Semicond. Technol. Sci.*, vol. 14, no. 4, pp. 419–426, Aug. 2014, doi: 10.5573/JSTS.2014.14.4.419.

[106] T. T. B. Nguyen, T. Nguyen Tan, and H. Lee, "Efficient QC-LDPC encoder for 5G new radio," *Electronics*, vol. 8, no. 6, p. 668, Jun. 2019, doi: 10.3390/electronics8060668.

[107] R. G. Gallager, “Low-density parity-check codes,” MIT Press, Cambridge, MA, USA, 1963.

[108] M. Sipser and D. A. Spielman, “Expander codes,” *IEEE Trans. Inform. Theory*, vol. 42, no. 6, pp. 1710–1722, Nov. 1996, doi: 10.1109/18.556667.

[109] J. Zhang and M. P. C. Fossorier, “A modified weighted bit-flipping decoding of low-density parity-check codes,” *IEEE Commun. Lett.*, vol. 8, no. 3, pp. 165–167.

[110] T. Wadayama, K. Nakamura, M. Yagita, Y. Funahashi, S. Usami, and I. Takumi, “Gradient descent bit flipping algorithms for decoding LDPC codes,” in *Proceedings on 2008 International Symposium on Information Theory and Its Applications*, Auckland, New Zealand, Apr. 2008, pp. 1–6. doi: 10.1109/ISITA.2008.4895387.

[111] D. V. Nguyen and B. Vasic, “Two-bit bit flipping algorithms for LDPC codes and collective error correction,” *IEEE Trans. Commun.*, vol. 62, no. 4, pp. 1153–1163, Apr. 2014, doi: 10.1109/TCOMM.2014.021614.130884.

[112] D. Oh and K. K. Parhi, “Low complexity decoder architecture for low-density parity-check codes,” *J. Sign. Process. Syst.*, vol. 56, pp. 217–228, 2009, doi: 10.1007/s11265-008-0231-5.

[113] I.-W. Yun, H. Lee, and J. T. Kim, “An alternative approach obtaining a normalization factor in normalized Min-Sum algorithm for low-density parity-check code,” *Wirel. Commun. Mob. Comput.*, vol. 2018, p. 1398191, Oct. 2018, doi: 10.1155/2018/1398191.

[114] K. Le Trung, F. Ghaffari, and D. Declercq, “An adaptation of Min-Sum decoder for 5G low-density parity-check codes,” in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, Sapporo, Japan, May 2019, pp. 1–5. doi: 10.1109/ISCAS.2019.8702344.

[115] Y. Liu, W. Tang, and D. G. M. Mitchell, “Efficient implementation of a threshold modified Min-Sum algorithm for LDPC decoders,” *IEEE Trans. Circuits Syst. II: Exp. Briefs*, vol. 67, no. 9, pp. 1599–1603, Sep. 2020, doi: 10.1109/TCSII.2020.3001601.

[116] X. Wu, M. Jiang, and C. Zhao, “Decoding optimization for 5G LDPC codes by machine learning,” *IEEE Access*, vol. 6, pp. 50179–50186, Sep. 2018, doi: 10.1109/ACCESS.2018.2869374.

[117] F. Guilloud, E. Boutillon, and J.-L. Danger, “λ-min decoding algorithm of regular and irregular LDPC codes,” in *Proceedings of the 3rd International Symposium on Turbo Codes and Related Topics*, Brest, France, Sep. 2003, pp. 451–454.

[118] V. Savin, “Self-corrected Min-Sum decoding of LDPC codes,” in *Proceedings on 2008 IEEE International Symposium on Information Theory*, Toronto, ON, Canada, Jul. 2008, pp. 146–150. doi: 10.1109/ISIT.2008.4594965.

[119] S. Lin and D. J. Costello, Jr., *Error control coding*, 2nd ed. Upper Saddle River, NJ, USA: Pearson Prentice Hall, 2004.

[120] D. Oh and K. K. Parhi, “Performance of quantized Min-Sum decoding algorithms for irregular LDPC codes,” in *Proceeednings on 2007 IEEE International Symposium on Circuits and Systems*, New Orleans, LA, USA, May 2007, pp. 2758–2761. doi: 10.1109/ISCAS.2007.378624.

[121] S. Myung, S.-I. Park, K.-J. Kim, J.-Y. Lee, S. Kwon, and J. Kim, “Offset and normalized Min-Sum algorithms for ATSC 3.0 LDPC decoder,” *IEEE Trans. on Broadcast.*, vol. 63, no. 4, pp. 734–739, Dec. 2017, doi: 10.1109/TBC.2017.2686011.

[122] G. Han and X. Liu, “An efficient dynamic schedule for layered belief-propagation decoding of LDPC codes,” *IEEE Commun. Lett.*, vol. 13, no. 12, pp. 950–952, Dec. 2009, doi: 10.1109/LCOMM.2009.12.091555.

[123] J. Zhang and M. P. C. Fossorier, “Shuffled iterative decoding,” *IEEE Trans. Commun.*, vol. 53, no. 2, pp. 209–213, Feb. 2005, doi: 10.1109/TCOMM.2004.841982.

[124] M. K. Roberts and R. Jayabalan, “An area efficient and high throughput multi-rate quasi-cyclic LDPC decoder for IEEE 802.11n applications,” *Microelectronics J.*, vol. 45, no. 11, pp. 1489–1498, Nov. 2014, doi: 10.1016/j.mejo.2014.07.003.

[125] S. Ajaz, T. T. B. Nguyen, and H. Lee, “An area-efficient half-row pipelined layered LDPC decoder architecture,” *J. Semicond. Technol. Sci.*, vol. 17, no. 6, pp. 845–853, Dec. 2017, doi: 10.5573/JSTS.2017.17.6.845.

[126] H. -J. Kang and B. -D. Yang, “Low-complexity, high-speed multi-size cyclic-shifter for quasi-cyclic LDPC decoder,” *Electron. lett.*, vol. 54, no. 7, pp. 452–454, Apr. 2018, doi: 10.1049/el.2017.4456.

[127] X. Chen, S. Lin, and V. Akella, “QSN—a simple circular-shift network for reconfigurable quasi-cyclic LDPC decoders,” *IEEE Trans. Circuits Syst. II: Exp. Briefs*, vol. 57, no. 10, pp. 782–786, Oct. 2010, doi: 10.1109/TCSII.2010.2067811.

[128] H.-J. Kang and B.-D. Yang, “Low-complexity multi-size cyclic-shifter for QC-LDPC codes,” *ETRI Journal*, vol. 39, no. 3, pp. 319–325, Jun. 2017, doi: 10.4218/etrij.17.0116.0341.

[129] Z. Wu and K. Su, “Updating conflict solution for pipelined layered LDPC decoder,” in *Proceedings on 2015 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, Ningbo, China, Sep. 2015, pp. 1–6. doi: 10.1109/ICSPCC.2015.7338879.

[130] H.-C. Lee, M.-R. Li, J.-K. Hu, P.-C. Chou, and Y.-L. Ueng, “Optimization techniques for the efficient implementation of high-rate layered QC-LDPC decoders,” *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 64, no. 2, pp. 457–470, Feb. 2017, doi: 10.1109/TCSI.2016.2612655.

[131] J. Jin and C. Tsui, “An energy efficient layered decoding architecture for LDPC decoder,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 8, pp. 1185–1195, Aug. 2010, doi: 10.1109/TVLSI.2009.2021479.

[132] C. Marchand and E. Boutillon, “LDPC decoder architecture for DVB-S2 and DVB-S2X standards,” in *Proceedings on 2015 IEEE Workshop on Signal Processing Systems (SiPS)*, Hangzhou, China, May 2015, pp. 1–5. doi: 10.1109/SiPS.2015.7345034.

[133] S. Yesil and M. Arslan, “Dual port ram based layered decoding for Multi Rate Quasi-Cyclic LDPC codes,” in *Proceedings on 2014 12th International Conference on Signal Processing (ICSP)*, Hangzhou, Zhejiang, China, Oct. 2014, pp. 1524–1530. doi: 10.1109/ICOSP.2014.7015253.

[134] V. A. Chandrasetty and S. M. Aziz, “Resource efficient LDPC decoders for multimedia communication,” *Integration*, vol. 48, pp. 213–220, Jan. 2015, doi: 10.1016/j.vlsi.2014.09.002.

[135] E. Amador, R. Pacalet, and V. Rezard, “Optimum LDPC decoder: a memory architecture problem,” in *Proceedings of the 46th Annual Design Automation Conference (DAC 2009)*, San Francisco, California, Jul. 2009, pp. 891–896. doi: 10.1145/1629911.1630141.

[136] X. Chen, J. Kang, S. Lin, and V. Akella, “Memory system optimization for FPGA-based implementation of quasi-cyclic LDPC codes decoders,” *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 58, no. 1, pp. 98–111, Jan. 2011, doi: 10.1109/TCSI.2010.2055250.

[137] R. Y. Shao, Shu Lin, and M. P. C. Fossorier, “Two simple stopping criteria for turbo decoding,” *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1117–1120, Aug. 1999, doi: 10.1109/26.780444.

[138] Y.-L. Ueng, Y.-L. Wang, C.-Y. Lin, J.-Y. Hsu, and Pangan Ting, “Modified layered message passing decoding with dynamic scheduling and early termination for QC-LDPC codes,” in *Proceedings on 2009 IEEE International Symposium on Circuits and Systems*, Taipei, Taiwan, May 2009, pp. 121–124. doi: 10.1109/ISCAS.2009.5117700.

[139] X. Zhao, Z. Chen, X. Peng, D. Zhou, and S. Goto, “A BER performance-aware early termination scheme for layered LDPC decoder,” in *Proceedings on 2010 IEEE Workshop On Signal Processing Systems*, San Francisco, CA, USA, Oct. 2010, pp. 416–419. doi: 10.1109/SIPS.2010.5624881.

[140] 3GPP TS 38.214 V16.5.0 (2021-03), *Technical Specification Group Radio Access Network; NR; Physical layer procedures for data (Release 16)*. 2021.

[141] S. Brkić and P. Ivaniš, “Analysis and design of LDPC codes for FTN signaling with finite number of turbo iterations,” in *Proceedings on 2019 27th Telecommunications Forum (TELFOR)*, Belgrade, Serbia, Nov. 2019, pp. 1–4. doi: 10.1109/TELFOR48224.2019.8971243.

[142] J. Nadal and A. Baghdadi, “Parallel and flexible 5G LDPC decoder architecture targeting FPGA,” *IEEE Trans. VLSI Syst.*, pp. 1–11, 2021, doi: 10.1109/TVLSI.2021.3072866.

[143] P. Hailes, L. Xu, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, “A survey of FPGA-based LDPC decoders,” *IEEE Commun. Surv. Tutorials*, vol. 18, no. 2, pp. 1098–1122, Quart 2016, doi: 10.1109/COMST.2015.2510381.

[144] X. Zhang and P. H. Siegel, “Quantized iterative message passing decoders with low error floor for LDPC codes,” *IEEE Trans. Commun.*, vol. 62, no. 1, pp. 1–14, Jan. 2014, doi: 10.1109/TCOMM.2013.112313.120917.

[145] A. Darabiha, A. C. Carusone, and F. R. Kschischang, “A bit-serial approximate min-sum LDPC decoder and FPGA implementation,” in *Proceedings on 2006 IEEE International Symposium on Circuits and Systems*, Island of Kos, Greece, 2006, p. 4. doi: 10.1109/ISCAS.2006.1692544.

[146] F. Angarita, J. Valls, V. Almenar, and V. Torres, “Reduced-complexity Min-Sum algorithm for decoding LDPC codes with low error-floor,” *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 61, no. 7, pp. 2150–2158, Jul. 2014, doi: 10.1109/TCSI.2014.2304660.

[147] I. Tsatsaragkos and V. Paliouras, “Approximate algorithms for identifying minima on Min-Sum LDPC decoders and their hardware implementation,” *IEEE Trans. Circuits Syst. II: Exp. Briefs*, vol. 62, no. 8, pp. 766–770, Aug. 2015, doi: 10.1109/TCSII.2015.2433451.

[148] C.-C. Cheng, J.-D. Yang, H.-C. Lee, C.-H. Yang, and Y.-L. Ueng, “A fully parallel LDPC decoder architecture using probabilistic Min-Sum algorithm for high-throughput applications,” *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 61, no. 9, pp. 2738–2746, Sep. 2014, doi: 10.1109/TCSI.2014.2312479.

[149] J. M. Català-Pérez, J. O. Lacruz, F. García-Herrero, J. Valls, and D. Declercq, “Second minimum approximation for Min-Sum decoders suitable for high-rate LDPC codes,” *Circuits Syst. Signal Process.*, vol. 38, no. 11, pp. 5068–5080, Nov. 2019, doi: 10.1007/s00034-019-01107-z.

Биографија аутора

Владимир (Лука) Петровић рођен је у Ужицу 21.11.1991. године. Основну школу и гимназију је завршио у Пожеги као ученик генерације. На Електротехничком факултету у Београду на Одсеку за електронику дипломирао је 27.06.2014. год. као студент генерације са просечном оценом 10, а мастер академске студије завршио је 25.09.2015. године такође са просечном оценом 10. Докторске академске студије на Електротехничком факултету уписао је 2015. године. На докторским студијама положио је све испите са оценом 10.

Од 3.11.2014. запослен је као сарадник у настави, а од 1.2.2016. као асистент на Катедри за електронику Електротехничког факултета у Београду. У настави је доминантно ангажован на предметима из области дигиталне електронике, дизајна VLSI система и дигиталне обраде сигнала. Учествовао је на више пројеката од којих су најзначајнији „Развој широкопојасног модема и интернет свича" у сарадњи са SANS R&D LLC из Сан Дијега (Калифорнија, САД), и „Сензор за детекцију људи", финансиран од стране Фонда за иновациону делатност РС, у сарадњи са NovelIC d.o.o из Београда.

Пре радног ангажовања на Електротехничком факултету током школске 2013/14. године био је на стручној пракси у „Aggios Europe d.o.o." у Београду, а током лета 2014. године на Универзитету Калифорније у Лос Анђелесу (*UCLA*).

Аутор је четири рада у међународним часописима (три категорије М21 и једног М22), два рада у часописима националног значаја (М51 и М53), четири рада на међународним конференцијама (М33) и једног рада на националној конференцији (М63). На конференцији *IcETRAN* 2016 добио је награду за најбољи рад младог аутора.

Области истраживања су му пројектовање дигиталних VLSI система и интегрисаних кола, дигитална обрада сигнала и хардверске реализације алгоритама обраде сигнала и алгоритама теорије информација.

Прилог 1.

Изјава о ауторству

Име и презиме аутора Владимир Петровић

Број индекса 2015/5034

Изјављујем

да је докторска дисертација под насловом

Флексибилни кодер и декодер кодова са проверама парности мале густине

- резултат сопственог истраживачког рада;
- да дисертација у целини ни у деловима није била предложена за стицање друге дипломе према студијским програмима других високошколских установа;
- да су резултати коректно наведени и
- да нисам кршио ауторска права и користио интелектуалну својину других лица.

Потпис аутора

У Београду, 21.9.2021.

Владимир Петровић

Прилог 2.

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора Владимир Петровић

Број индекса 2015/5034

Студијски програм Електроника

Наслов рада Флексибилни кодер и декодер кодова са проверама парности мале густине

Ментор др Лазар Сарановац, редовни професор

Изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао ради похрањивања у **Дигиталном репозиторијуму Универзитета у Београду**.

Дозвољавам да се објаве моји лични подаци везани за добијање академског звања доктора наука, као што су име и презиме, година и место рођења и датум одбране рада.

Ови лични подаци могу се објавити на мрежним страницама дигиталне библиотеке, у електронском каталогу и у публикацијама Универзитета у Београду.

Потпис аутора

У Београду, 21.9.2021.

Владимир Петровић

Прилог 3.

Изјава о коришћењу

Овлашћујем Универзитетску библиотеку „Светозар Марковић“ да у Дигитални репозиторијум Универзитета у Београду унесе моју докторску дисертацију под насловом:

Флексибилни кодер и декодер кодова са проверама парности мале густине

која је моје ауторско дело.

Дисертацију са свим прилозима предао сам у електронском формату погодном за трајно архивирање.

Моју докторску дисертацију похрањену у Дигитални репозиторијум Универзитета у Београду и доступну у отвореном приступу могу да користе сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons) за коју сам се одлучио.

1. Ауторство (CC BY)
2. Ауторство – некомерцијално (CC BY-NC)
3. Ауторство – некомерцијално – без прерада (CC BY-NC-ND)
4. Ауторство – некомерцијално – делити под истим условима (CC BY-NC-SA)
5. Ауторство – без прерада (CC BY-ND)
6. Ауторство – делити под истим условима (CC BY-SA)

(Молимо да заокружите само једну од шест понуђених лиценци.
Кратак опис лиценци дат је саставни део ове изјаве).

Потпис аутора

У Београду, 21.9.2021.

Владимир Петровић

1. **Ауторство**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце, чак и у комерцијалне сврхе. Ово је најслободнија од свих лиценци.

2. **Ауторство – некомерцијално.** Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела.

3. **Ауторство – некомерцијално – без прерада**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела. У односу на све остале лиценце, овом лиценцом се ограничава највећи обим права коришћења дела.

4. **Ауторство – некомерцијално – делити под истим условима**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца не дозвољава комерцијалну употребу дела и прерада.

5. **Ауторство – без прераде**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца дозвољава комерцијалну употребу дела.

6. **Ауторство – делити под истим условима**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца дозвољава комерцијалну употребу дела и прерада. Слична је софтверским лиценцама, односно лиценцама отвореног кода.