New challenges in designing polar code decoders for 5G

Oualid Mouhoubi

DOCTORAT BRETAGNE LOIRE MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPERIEURE MINES-TELECOM ATLANTIQUE
BRETAGNE PAYS DE LA LOIRE - IMT ATLANTIQUE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Télécommunications*

Par
Oualid MOUHOUBI

Nouveaux défis dans la conception de décodeurs de codes polaires pour la 5G

Thèse présentée et soutenue à IMT Atlantique, Brest, France, le 23 Septembre 2022
Unité de recherche : Lab-STICC
Thèse N° : 2022IMTA0298

Rapporteurs avant soutenance :

Fabrice MONTEIRO Professeur, Université de Lorraine
Vahid MEGHDADI Professeur, Université de Limoges/ENSIL-ENSCI

Composition du Jury :

Président :	**Catherine DOUILLARD**	Professeur, IMT Atlantique
Examinateurs :	**Fabrice MONTEIRO**	Professeur, Université de Lorraine
	Vahid MEGHDADI	Professeur, Université de Limoges/ENSIL-ENSCI
	Camille LEROUX	Maître de conférences, Bordeaux INP/ENSEIRB-MATMECA
Dir. de thèse :	**Amer BAGHDADI**	Professeur, IMT Atlantique
Encadrant :	**Charbel ABDEL NOUR**	Maître de conférences, HDR, IMT Atlantique
Invité :	**David GNAEDIG**	CTO, TurboConcept

Summary (English)

Proposed in the last few years, polar codes represent one of the latest additions to the family of forward error correction (FEC) codes. They have been adopted as the coding scheme in the control channel of the 3rd Generation Partnership Project (3GPP) New Radio (NR) standard for the fifth generation of cellular mobile communications (5G). However, the challenging requirements introduced by the 5G control channel in terms of block length and code rate flexibility render unsuitable most of the previously published hardware polar decoder implementations. Indeed, these latter focused mainly on successive cancellation decoders with high throughput, limited flexibility and error correction capabilities. With stringent constraints on end-to-end delay and error correction, the 5G NR steers towards low-latency list-based decoder architectures. In this context, several original contributions are proposed in this thesis work. The first major contribution is related to design space exploration and concerns the study of the impact of main code and decoder design parameters on the latency, throughput, and the hardware complexity of semi-parallel decoding architectures. The impact of these parameters on the hardware efficiency of semi-parallel architectures is significant. Therefore, we propose two multi-frame decoding approaches that increase the throughput and improve the utilisation rate of the processing units of these architectures. Detailed analytical and logic synthesis results are provided and compared for a large range of values in order to constitute a reference for the implementation of flexible, yet efficient FEC decoders for polar codes. Furthermore, a complete software simulation environment of polar coding/decoding is proposed for performance evaluation under different algorithms in both floating-point and fixed-point data representation. The second major contribution concerns the design of an original flexible and low-latency list-based hardware architecture for decoding 5G NR polar codes. This was achieved based on the design space exploration study, and motivated by the need to provide a hardware-efficient polar decoder that supports the required flexibility and latency levels for 5G NR. The proposed design supports all the frame sizes and code rates defined in 3GPP with throughput and latency values meeting the standard requirements. Furthermore, the decoder architecture has been extended to support the proposed multi-frame decoding scheme, particularly suited for blind decoding of downlink control information.

Résumé (Français)

Proposés ces dernières années, les codes polaires représentent l'un des derniers apports à la famille des codes correcteurs d'erreurs (FEC). Ils ont été adoptés comme schéma de codage pour le canal de contrôle de la norme 5G NR (New Radio) pour la cinquième génération de communications mobiles cellulaires. Cependant, les exigences élevées introduites par le canal de contrôle de la 5G en termes de flexibilité de la longueur de code et du rendement de codage font que la plupart des décodeurs matériels de codes polaires publiés antérieurement sont inadaptés. En effet, ces derniers se sont principalement focalisés sur des décodeurs à annulation successive offrant un débit élevé, une flexibilité limitée et des pouvoirs de correction d'erreurs réduits. Avec des contraintes strictes sur le délai de bout en bout et sur la correction d'erreurs, la 5G NR nécessite des architectures de décodeurs à liste à faible latence. Dans ce contexte, plusieurs contributions originales sont proposées dans ce travail de thèse. La première contribution majeure est liée à l'exploration de l'espace de conception et concerne l'étude de l'impact des principaux paramètres du code et du décodeur sur la latence, le débit et la complexité matérielle des architectures de décodage semi-parallèles. L'impact de ces paramètres sur l'efficacité matérielle des architectures semi-parallèles est important. Par conséquent, nous proposons deux approches de décodage multi-trames qui augmentent le débit et améliorent le taux d'utilisation des unités de traitement de ces architectures. Des résultats analytiques détaillés et des résultats de synthèse logique sont fournis et comparés pour une large gamme de valeurs afin de constituer une référence pour la mise en œuvre de décodeurs FEC flexibles mais efficaces pour les codes polaires. Par ailleurs, un environnement logiciel complet de simulation du codage/décodage de codes polaires est proposé pour évaluer les performances de différents algorithmes dans une représentation des données en virgule flottante et en virgule fixe. La deuxième contribution majeure concerne la conception d'une architecture matérielle originale, flexible et à faible latence, basée sur le décodage à liste des codes polaires de la 5G NR. Ce résultat a été obtenu sur la base de l'étude d'exploration de l'espace de conception, et motivé par le besoin de fournir un décodeur polaire efficace qui supporte les niveaux de flexibilité et de latence requis pour le standard. Le décodeur proposé supporte toutes les tailles de trame et tous les rendements de code définis dans la 5G avec des valeurs de débit et de latence conformes aux exigences de la norme. En outre, l'architecture du décodeur a été étendue pour supporter le schéma de décodage multi-trame proposé, particulièrement adapté au décodage aveugle des informations de contrôle de la liaison descendante.

Introduction

L'énorme croissance de la connectivité et la demande grandissante de trafic de données, de vidéo et de messagerie qui ont conduit au développement de la quatrième génération de téléphonie mobile (4G), n'ont cessé de se développer. En outre, le besoin d'une fiabilité supérieure et d'une latence réduite où les réseaux de communication mobile devraient prendre en charge des milliards d'appareils connectés, a poussé la 4G/LTE à ses limites et a motivé le développement de la norme de cinquième génération de téléphonie mobile (5G).

La nouvelle vague de révolution technologique est la raison de l'émergence de nouvelles applications et de nouveaux services tels que l'intelligence artificielle, la maison intelligente, les véhicules autonomes, les systèmes de livraison par drones, les villes intelligentes, les usines intelligentes, etc. Les services de réseaux mobiles ont été classés en trois scénarios d'utilisation de la 5G pour 2020 et au-delà par l'Union Internationale des Télécommunications, à savoir:

— Enhance Mobile BroadBand (eMBB) : il s'agit d'une évolution continue du service haut débit mobile classique (MBB) de la 4G/LTE. Il implique des cas d'utilisation axés sur les données et nécessitant des débits de données élevés, ce qui se traduit par une expérience utilisateur plus rapide et de meilleure qualité de service.

— Ultra Reliable Low Latency Communication (uRLLC) : ce service est soumis à des exigences strictes en matière de latence et de fiabilité et vise à prendre en charge des applications critiques dans les domaines de la fabrication, de la transmission d'énergie, des transports et des soins de santé.

— Massive Machine Type Communications (mMTC) : ce service concerne les applications industrielles et IoT et vise à fournir l'accès au réseau sans fil à un grand nombre de dispositifs qui échangent des informations et génèrent des données via de petits paquets.

Avec tous ces services nécessitant de coexister dans un environnement sans fil unique, la 5G doit répondre à un large éventail de performance clés tels que la faible latence, le haut débit et le faible coût/consommation d'énergie. Afin de répondre à ces nouvelles exigences de la 5G, le projet de partenariat de troisième génération (3GPP) New Radio (NR) adopte et spécifie un ensemble de techniques avancées.

Les codes correction d'erreur (ECC) sont considérés comme l'un des composants clés de la technologie 5G NR. Deux nouveaux schémas de codage de canal ont été sélectionnés

au cours du processus de normalisation de la 5G. Les codes de contrôle de parité à faible densité (LDPC) ont été adoptés comme schéma de codage des données. Ils sont conçus pour supporter un débit élevé, un rendement de code et une longueur de code variables ainsi qu'une Hybrid Automatic Repeat Request (HARQ), en plus d'une très bonne capacité de correction des erreurs. D'autre part, les codes polaires ont été adoptés comme un nouveau schéma de codage pour les informations de contrôle et sont conçus pour produire une bonne performance de correction d'erreurs avec des longueurs de bloc courtes et des rendement de code très variés tout en imposant des contraintes strictes sur la latence de décodage [27]. Dans ce contexte, le travail de thèse présenté est dédié à l'étude et à l'exploration des techniques de décodage des codes polaires dans le but de proposer des solutions de décodage efficaces pour les codes polaires de la 5G NR.

Au début de ce travail de thèse, l'état de l'art manquait de publications relatives aux décodeurs de codes polaires qui répondent conjointement aux exigences de la 5G NR en termes de performance de correction d'erreur, de latence et de flexibilité en termes de longueur de bloc et de rendement de code. Cependant, au cours de la thèse, la communauté scientifique a constamment amélioré les algorithmes de décodage, en proposant de nouvelles techniques ainsi que de nouvelles conceptions de décodeurs matériels. Pourtant, les exigences strictes introduites sur le canal de contrôle de la 5G en termes de longueur de bloc et de flexibilité liée au rendement de code rendent inadaptés la plupart des décodeurs de codes polaires matériels déjà publiés à ce moment-là. Ceci motive notre travail de thèse en proposant une architecture matérielle flexible originale pour le décodage des codes polaires de la 5G NR et qui peut supporter toutes les tailles de trame et les rendements de code définis dans la norme avec une faible latence de décodage. L'approche utilisée dans le cadre de ce travail est d'abord de mener une étude complète sur les codes polaires et leurs algorithmes de décodage, puis d'effectuer une exploration approfondie de l'espace de conception des architectures de décodeurs de codes polaires pour enfin proposer une nouvelle architecture matérielle efficace ciblant les technologies FPGA. Motivés par l'importance du décodage aveugle dans le canal de contrôle 5G NR, nous étudions le décodage de multiples trames dans le but de prendre en charge efficacement cette fonctionnalité également.

Plan du manuscrit et contributions

Ce manuscrit de thèse commence par présenter les concepts de base liés aux codes polaires, y compris leur processus d'encodage et de décodage, en mettant l'accent sur les

codes polaires adoptés dans la norme 5G NR. Dans un premier temps, le principe de la polarisation des canaux est introduit. Ensuite, l'algorithme de décodage par annulation successive (SC) et son extension basée sur le décodage par liste (SCL) sont décrits, suivis d'un bref aperçu des autres formes de décodage des codes polaires. Suite à cela, des variantes simplifiées de décodage par annulation successive appliquant ce que l'on appelle l'élagage des arbres sont présentées comme des techniques de réduction de la latence et d'augmentation du débit. Enfin, un aperçu des codes polaires adoptés dans la norme 5G NR ainsi que leur schéma d'encodage est présenté.

Les exigences strictes introduites par la norme 5G en termes de longueur de bloc et de flexibilité liée au rendement de code, ainsi qu'une faible latence de bout en bout et des performances élevées en matière de correction d'erreurs représentent un défi majeur pour leur mise en œuvre matérielle. Afin d'adresser ce problème, les contributions de ce travail de thèse ont été divisées en deux parties principales. Dans la première partie une exploration approfondie de l'espace de conception des architectures de décodeurs polaires a été menée avec de plusieurs contributions.

Une étude sur les performances de correction d'erreurs des codes polaires 5G NR en termes de taux d'erreurs binaires (BER) et de taux d'erreurs de trames (FER) a d'abord été menée. En effet, les performances des algorithmes simplifiés basés sur des techniques d'élagage de l'arbre de décodage des décodeurs SCL sont évaluées et comparées aux performances du décodeur SCL classique pour une taille de liste de huit. Pour éviter que les performances de correction d'erreurs des codes polaires ne se dégradent au niveau de la conception matérielle, plusieurs niveaux de quantification pour le passage des messages et les métriques internes, à savoir les Log Likelihood Ratios (LLR) et les Path Métriques (PM), sont étudiés. Pour cela, les performances de simulation du décodeur en virgule fixe et en virgule flottante sont comparées.

Un simulateur logiciel dédiée aux codes polaires a été développé et présenté dans le chapitre 3. Le simulateur inclut l'ensemble de la chaîne d'encodage et de décodage, depuis l'allocation des canaux binaires, l'insertion des bits de CRC (contrôle de redondance cyclique) et les schémas d'entrelacement jusqu'à l'adaptation du rendement (rate-matching) telle que définie dans la 5G NR, ainsi que leurs opérations inverses pour assurer le processus de la phase de décodage. Grâce à la complétude du simulateur logiciel, les performances en termes de taux d'erreurs binaires des codes polaires 5G NR sous une diversité d'algorithmes de décodages sont évaluées en virgule flottante mais aussi en virgule fixe, fournissant ainsi un outil efficace pour sélectionner le meilleur schéma de quantification pour les décodeurs

polaires matériels.

Étant donné que l'algorithme de décodage SC augmentée par liste est le mieux adapté pour décoder les codes polaires du canal de contrôle 5G NR, principalement grâce à ses bonnes performances de correction d'erreurs notamment lorsqu'il est assisté par des codes externes tels que le CRC et des bits de parité. Cependant, il n'est pas facile de trouver l'architecture matérielle la mieux adaptée au décodage, particulièrement en raison des exigences strictes de conformité aux flexibilités de longueurs de bloc et de rendements de code, tout en maintenant une latence de décodage et une complexité matérielle faibles. Dans ce contexte, une exploration de l'espace de conception des décodeurs de codes polaires a été menée. Par conséquent, une analyse détaillée de l'impact des principaux paramètres de conception de code et du décodeur sur la latence, le débit, la complexité matérielle et l'efficacité matérielle des architectures de décodage de codes polaires, en particulier ceux de la 5G NR est fourni. Dans ce contexte, le modèle d'architecture semi-parallèle s'avère être le plus adapté grâce à une plus grande flexibilité algorithmique et architecturale au niveau de la conception. Par conséquent, sur la base d'une étude analytique détaillée et des résultats de synthèse logique, la latence, le débit et la complexité du décodeur ont été évalués pour de multiples variantes d'algorithmes de décodage SCL et pour un nombre variable d'éléments de traitement (processing element). Les résultats ont montré que les conceptions flexibles en termes de longueur de code et de rendement de code limitent l'avantage d'augmenter le nombre d'éléments de traitement et préconisent de définir divers types de codes spéciaux au sein de l'arbre de décodage des codes polaires tout en augmentant le niveau d'élagage de l'arbre. En effet, si l'utilisation d'un grand nombre d'éléments de traitement permet une réduction significative de la latence pour les codes polaires de longueur élevée, cet avantage devient négligeable pour les codes polaires de petites longueurs, ce qui pénalise l'efficacité matérielle du décodeur. En outre, certains types de codes constitutifs spéciaux sont plus susceptibles d'apparaître à des rendements de code faibles, comme les codes à rendement nul (codes R0) et les codes à répétitions (codes REP), tandis que d'autres sont plus susceptibles d'apparaître à des rendements de codes élevés, comme les codes spéciaux de rendement égale à 1 (codes R1) et les code de contrôle à parité unique (SPC). De plus, la longueur de ces codes constitutifs spéciaux en nombre de bits impliqués diminue avec la taille de la trame du code polaire. Cela peut avoir un impact important sur les mesures d'efficacité de l'implémentation matérielle. Par conséquent, de multiples compromis entre les paramètres algorithmiques et architecturaux peuvent être tirés de ces résultats. À cet égard, nous avons proposé deux approches de

décodage multi-trame augmentant le débit et améliorant l'activité des unités de traitement au prix de ressources mémoire et d'une latence supplémentaire.

Dans la deuxième partie et sur la base des résultats obtenu dans le chapitre 3, de multiples choix d'implémentation de décodeurs de codes polaires se présentent et un large spectre de compromis complexité/performance est fourni. Ainsi, ces résultats sont utilisés pour proposer dans le chapitre 4 une architecture matérielle originale pour le décodage des codes polaires 5G NR des deux canaux de contrôle de la liaison montante (PUCCH) et de la liaison descendante (PDCCH). L'architecture de décodage en *top-level* et présentée en début du chapitre et les différents éléments qui composent l'architecture sont détaillés au fur et à mesure. L'architecture de décodage basé sur l'algorithme à annulation successive nécessite le stockage en mémoire de plusieurs types de données. Une structure de mémoire adaptée à la nature séquentielle de l'algorithme de décodage à annulation successive est proposée et détaillée pour chaque type de mémoire utilisée. Par exemple, trois types de gestion de la mémoire des données LLR sont présentés afin d'assurer un acheminement cohérent des données LLRs entre les éléments de traitement et la mémoire de données et de maintenir une latence de décodage minimale. Au lieu de décoder les noeuds spéciaux individuellement, nous avons proposé de concevoir une architecture de décodeur des nœuds spéciaux qui regroupe les opérations communes effectuées par les différents nœuds spéciaux. Ensuite, un module d'identification dynamique des noeuds spéciaux est proposé afin que le décodeur puisse continuer de bénéficier des techniques d'élagage d'arbre pour accélérer le décodage tout en maintenant la conformité avec la 5G NR et les différentes combinaisons définies de rendement de de longueur de code. Enfin, d'autres contributions en relation directe avec le décodage de plusieurs bits en parallèle sont présenté dans ce chapitre notamment le calcul de sommes partielles et le calcul des bits de CRC.

Le décodeur proposé a ensuite été décrit en VHDL, validé et synthétisé pour la technologie cible FPGA. Une analyse approfondie des principales performances clés du décodeur, y compris la latence de décodage, le débit et l'utilisation des ressources du FPGA, a été fournie et des comparaisons avec les implémentations FPGA de code polaire les plus récentes ont été effectuées. En effet, les valeurs mesurées de débit et de latence du décodeur proposé obtenues avec une cible FPGA sont capables de répondre aux exigences de la norme de la 5G. De plus, les résultats de synthèse ont montré une efficacité matérielle qui se compare favorablement aux implémentations FPGA de codes polaires de l'état de l'art. La conception du décodeur que nous proposons réduit la latence et la complexité du décodage par rapport au décodeur polaire de Xilinx récemment disponible, et le seul

décodeur publié et entièrement conforme à la norme 5G NR. Enfin, nous avons proposé une nouvelle façon de décoder plusieurs trames de codes polaires simultanément sur la base de l'architecture du décodeur proposée avec seulement une modification mineure qui préserve toujours la flexibilité du décodeur. Une implémentation de cette technique a été réalisée et les résultats de synthèse logique ont été discutés.

Conclusion et perspectives

Le travail présenté dans ce manuscrit visait à étudier l'impact des principaux paramètres de conception de code et du décodeur sur la latence, le débit, la complexité matérielle et l'efficacité matérielle des architectures de décodage semi-parallèles, ainsi qu'à proposer et à implémenter une architecture matérielle originale pour le décodage des codes polaires 5G NR.

Dans le chapitre 2, les concepts de base des codes polaires ont été rappelés, ainsi qu'une présentation détaillée de l'algorithme de décodage par annulation successive et des différentes variantes de cet algorithme récemment proposées dans la littérature pour améliorer les performances de décodage. Un accent particulier a été mis dans ce chapitre sur les techniques d'élagage des arbres, motivées par leur impact positif sur la latence et le débit. Le chapitre a également introduit la spécification des codes polaires adoptés dans les 5G NR, nécessaire à la compréhension des contributions proposées dans les chapitres suivants.

Dans le chapitre 3, le modèle de simulation développé pour intégrer le schéma complet d'encodage et de décodage des codes polaires de la 5G NR est présenté, en plus du canal de communication et des différentes opérations d'entrelacement et d'adaptation de rendement (rate-matching). Ce simulateur logiciel a été utilisé pour évaluer les performances des codes polaires sous différents types d'algorithmes SC de décodage rapide et a été utilisé pour évaluer les performances de l'architecture du décodeur avec différents paramètres liés en particulier à la quantification. L'espace de conception du décodeur semi-parallèle de codes polaires a été exploré en termes de choix de parallélisme, d'algorithme et de complexité matérielle. Les niveaux de conception algorithmique et architecturale ont été pris en considération, ainsi que diverses techniques d'élagage des arbres, afin d'évaluer la relation entre la complexité matérielle et les principales performances clés des décodeurs polaires. En particulier, les performances de latence et de débit des décodeurs basés sur le décodeur SC ont été ciblées. De plus, l'activité des différentes unités de traitement des

décodeurs basés sur le décodeur SC, y compris les éléments de traitement et les décodeurs de nœuds spéciaux, a été évaluée.

Dans le chapitre 4, nous avons proposé une architecture matérielle originale pour le décodage des codes polaires spécifiés dans la 5G NR pour le canal de contrôle des liaisons physiques montantes et descendantes. Nous avons détaillé les différents éléments qui composent l'architecture proposée, à savoir la structure mémoire, les unités de calcul et les réseaux de permutation. Un décodeur original des nœuds spéciaux dans le cadre d'un décodage par liste, un calcul multi-bits de sommes partielles et un contrôle CRC (adapté au décodage multi-bits) ont également été proposés et détaillés dans ce chapitre. Grâce à un identificateur des nœuds spéciaux en temps réel, le décodeur proposé continue de bénéficier des techniques d'élagage d'arbre pour accélérer le décodage tout en maintenant la conformité avec les exigences de la norme 5G NR et les diverses combinaisons définies de rendements et de longueurs de code. Le décodeur proposé a été décrit en VHDL, validé et synthétisé pour la technologie cible FPGA. Une analyse approfondie des principales performances clés du décodeur, y compris la latence de décodage, le débit et l'utilisation des ressources du FPGA, a été fournie et des comparaisons avec des implémentations FPGA de code polaire ont été effectuées. En effet, les valeurs mesurées de débit et de latence du décodeur proposé obtenues avec une cible FPGA sont capables de répondre aux exigences de la 5G. De plus, les résultats de synthèse ont montré une efficacité matérielle qui se compare favorablement aux implémentations FPGA de codes polaires de l'état de l'art. La conception du décodeur que nous proposons réduit la latence et la complexité du décodage par rapport au décodeur polaire de Xilinx récemment disponible, et le seul décodeur publié et entièrement conforme à la norme 5G NR. Enfin, nous avons proposé une nouvelle méthode pour décoder simultanément plusieurs trames de code polaire. Bien qu'elle soit basée sur l'architecture du décodeur proposée, elle n'introduit qu'une modification mineure qui préserve toujours la flexibilité du décodeur. Une implémentation simple de cette technique a été réalisée et les résultats ont été discutés.

Les recherches menées dans le cadre de ce travail de thèse ont montré qu'il était difficile de satisfaire simultanément les contraintes de latence de décodage, de flexibilité extrême en termes de taille de trame et de rendement de code, de performances de décodage et d'implémentation efficace, compte tenu de la courte histoire des codes polaires. Cependant, motivée par leur adoption dans la 5G, la communauté scientifique a fait et fait encore des progrès importants à cet égard. Ce travail constitue une contribution à cet effort. En effet, l'architecture proposée est capable de fonctionner avec une latence dans les ordres

du μs avec une complexité de décodage relativement faible et peut donc s'adapter aux contraintes strictes requises par la norme 5G qui a sélectionné les codes polaires pour couvrir le canal de contrôle. Cependant, des extensions et des améliorations sont encore possibles en ce qui concerne le travail effectué. Nous citons ci-dessous quelques suggestions d'études qui peuvent être menées :

1. Une voie à suivre suite aux résultats du chapitre 3 est d'étudier et d'analyser l'impact du décodage d'autres types de nœuds spéciaux [40] et des nœuds spéciaux généralisés [22] sur la latence, la complexité et le débit. L'analyse menée dans ce chapitre a ciblé l'ensemble des codes polaires 5G NR. Par conséquent, des paramètres de conception de code et de décodeur limités ont été considérés. Ainsi, les travaux futurs pourraient prendre en considération des tailles de liste variables, des codes polaires plus grands et différents ensembles de constructions de codes polaires. En outre, d'autres algorithmes de décodage pourraient être envisagés pour une comparaison et une analyse plus larges.
2. Dans les dernières parties du Chapitre 3, nous avons étudié le décodage simultané de plusieurs trames de codes polaires en proposant deux approches efficaces de décodage multi-trames. Une implémentation simple pour le décodage parallèle de deux trames, qui utilise alternativement les mêmes éléments de traitement et nœuds spéciaux, du même code polaire a été conçue. L'étape suivante consisterait à concevoir une nouvelle architecture de décodage multi-trames mettant en œuvre les algorithmes très rapides basés sur le décodage SC. Une telle architecture basée sur la deuxième approche présentée au chapitre 3 consistant à dupliquer le nombre d'éléments de traitement tout en utilisant un décodeur de nœuds spéciaux unique augmenterait de manière importante le débit du décodeur tout en bénéficiant de la flexibilité fournie par le décodeur proposé au chapitre 4. En outre, l'extension de cette architecture pour prendre en charge le décodage simultané de plusieurs trames de différentes longueurs de bloc et de différents rendements de code pourrait également être étudiée.
3. Les deux approches de décodage multi-trame ont été introduites dans le but d'augmenter le débit et l'efficacité matérielle du décodeur. De plus, une telle approche est très utile pour le décodage aveugle des codes polaires de la liaison physique descendante (PDCCH) [23, 69]. En effet, 44 candidats sont décodés au niveau de l'équipement utilisateur (UE) pour identifier l'information de contrôle de liaison descendante (DCI) portant l'information de contrôle de l'UE [1]. Cependant, au niveau du récepteur, l'UE n'a pas besoin de décoder les candidats PDCCH qui

appartiennent à l'espace de recherche avec l'algorithme SCL. Par conséquent, une étude de recherche consisterait à examiner l'utilisation du décodeur SCL à liste proposé ($L = 8$) en tant que L décodeurs SC (L-SC) afin de réduire l'espace de recherche des candidats dans une première phase du décodage aveugle [24]. Ensuite, le décodeur SCL peut être utilisé pour décoder un nombre réduit de candidats dans la deuxième phase du décodage.

Acknowledgement

Young of 24 years, I was fascinated by the digital electronics field and especially the design of hardware architectures and circuits dedicated to implement specific and complex functions. Indeed, Forward Error Correcting codes are one of the most complex topics to address from both algorithmic and design point of view.

With these considerations, I found a Master's level internship on Reed-Muller codes in Brest, and 4 years later I am a doctor of philosophy, who dedicated the last words of his manuscript to acknowledge the people who made this adventure possible, rewarding and successful. First of all, I would like to thank my dad Abdelkader and my mom Nadira for their support and all the efforts they have made to allow me to receive a good quality education since my first year of schooling until the final day of my thesis. Special thanks to my advisors Amer Baghdadi, Charbel Abdel Nour and David Gnaedig for their guidance, their efforts and all the advice they gave me during this work. And certainly for teaching me how to work. I sincerely believe that thanks to your work I have become a different person. Thank you for agreeing to work with me, I have learned a lot from your great knowledge.

A great thank is also addressed to my sister Nihad, my brother Imad and the people I was interacting with on a daily basis, including lab colleagues especially Jérémy, Rami and Khaled and friends Said, Réda, Ghouti, Aomar, Mounia, Mouna, Mokrane, Youssef, Mehdi, Hamza, Amine, all ES Locmaria team, and all the people I didn't mention here.

At the end of the journey, obtaining a Ph.D. degree is a pleasant prize for long hours, weeks and years of exploration that led to a mental (and physical) growth. Every single detail, every work meeting, every deception/achievement of this adventure was an increment, a positive perturbation that contributed to building the person I am today.

Table of Contents

List of Figures

List of Tables

List of Acronyms

3GPP	Third Generation Partnership Project.
4G	4th generation of cellular mobile communications.
5G	5th generation of cellular mobile communications.
ALM	Adaptive Logic Module.
ASIC	Application Specific Integrated Circuit.
AWGN	Additive White Gaussian Noise.
B-DMC	Binary Discrete Memoryless Channel.
BER	Bit Error Rate.
BP	Belief Propagation.
BPSK	Binary Phase-Shift Keying.
BRAM	Block Random Access Memory.
BSC	Binary Symmetric Channel.
CASCL	CRC-Aided Successive Cancellation List.
CRC	Cyclic Redundancy Check.
DCI	Downlink Control Information.
ECC	Error Correction Codes.
eMBB	enhanced Mobile BroadBand.
FASCL	Fully Adaptive Successive Cancellation List.
Fast SSCL	Fast Simplified Successive Cancellation List.
Fast SSCL-SPC	Fast Simplified Successive Cancellation List- Single Parity Check.
FER	Frame Error Rate.

FF	Flip-Flop.
FPGA	Field-Programmable Gate Array.
FSM	Finite State Machine.
HARQ	Hybrid Automatic Repeat reQuest.
IoT	Internet of Things.
ITU	International Telecommunication Union.
LDPC	Low Density Parity Check.
LLR	Log Likelihood Ratio.
LTE	Long Term Evolution.
LUT	Look-Up Table.
ML	Maximum-Likelihood.
mMTC	Massive Machine Type Communications.
NR	New Radio.
PASCL	Partially Adaptive Successive Cancellation List.
PBCH	Physical Broadcast CHannel.
PC	Parity Check.
PDCCH	Physical Downlink Control CHannel.
PE	Processing Element.
PM	Path Metric.
PS	Partial Sum.
PSN	Partial Sum Network.
PU	Processing Unit.
PUCCH	Physical Uplink Control CHannel.
PUSCH	Physical Uplink Shared CHannel.
REP	Repetition.
RNTI	Radio Network Temporary Identifier.
SC	Successive Cancellation.
SCAN	Soft CANcellation.
SCF	Successive Cancellation Flip.
SCL	Successive Cancellation List.
SCS	Successive Cancellation Stack.
SM	Sign and Magnitude.
SNLD	Special Node List Decoder.
SNR	Signal-to-Noise Ratio.

SPC	Single Parity Check.
SSC	Simplified Successive Cancellation.
SSCL	Simplified Successive Cancellation List.
SSCL-SPC	Simplified Successive Cancellation List- Single Parity Check.
UCI	Uplink Control Information.
URLLC	Ultra-Reliable Low-Latency Communication.
VHDL	VHSIC Hardware Description Language.
XOR	eXclusive-OR.

Chapter 1

Introduction

Context of the 5G communication standard and services

The tremendous amount of growth in connectivity and the increasing demand for data, video and messaging traffic that led to the 4th Generation mobile communication standard (4G) [83, 88], has continued to expand. Furthermore, the need for superior reliability and reduced latency for the massive internet of things (IoT) ecosystem [21, 80] where networks are expected to support billions of connected devices has pushed 4G/LTE to its limits and motivated the development of the 5th Generation of mobile communications standard (5G) [12, 74].

The new wave of technological revolution is the reason of the emergence of new applications and services such as artificial intelligence, smart home, autonomous vehicles, drone-based delivery systems, smart cities, smart factories, etc. Mobile network services have been categorized into three 5G usage scenarios [2] for 2020 and beyond by the International Telecommunication Union (ITU) radio communications [6]:

— Enhanced Mobile Broadband (eMBB) service: which is a continuing evolution of the conventional Mobile Broadband (MBB) service of 4G/LTE. It involves data-driven use cases requiring high data rates, resulting in a faster and better user experience.

— Ultra-reliable and Low-latency Communications (uRLLC) service: which has strin-

gent requirements on latency and reliability and aims to support mission critical applications in the areas of manufacturing, energy transmission, transportation and healthcare.

— Massive Machine Type Communications (mMTC) service: which concerns industrial and IoT applications and aims to provide the access to the wireless network for a large number of devices that exchange information and generate data via small packets per connection.

With all these services needing to coexist in a unique wireless environment, 5G is expected to cope with a broad range of various key performance indicators such as low latency, high throughput and low cost/energy consumption. In order to address these 5G new requirements, the third Generation Partnership Project (3GPP) New Radio (NR) is adopting and specifying a set of advanced enabling techniques.

Error Correction Codes (ECC) are considered as one of the key components of 5G NR technology. Two new capacity-approaching channel coding schemes have been selected during the 3GPP standardization process. Low-density parity-check (LDPC) codes have been adopted as the coding scheme for data. They are designed to support high throughput, a variable code rate and length and hybrid automatic repeat request in addition to good error correcting capability [39]. On the other hand, polar codes have been adopted as a new coding scheme for control information and are designed for producing good error correcting performance [47, 75] under short block length and various code rates while placing stringent constraints on the decoding latency [27]. In this context, the presented PhD thesis work is dedicated to the study and exploration of polar decoding techniques in the aim of proposing efficient decoder solutions for the 5G NR polar codes.

Motivation of the thesis

The design criteria [27] of the 5G NR polar codes include multiple key performance indicators and are summarized in Fig. 1.1.

At the starting of this PhD work, the state-of-the-art was lacking publications related to polar decoders that jointly meet the 5G NR requirements in terms of error correcting performance, latency, and flexibility in frame size and code rate. However, during the thesis time, the scientific community has consistently improved the decoding algorithms, proposing new techniques as well as new hardware decoder designs. Yet, the challenging requirements introduced by the 5G control channel in terms of block length and code

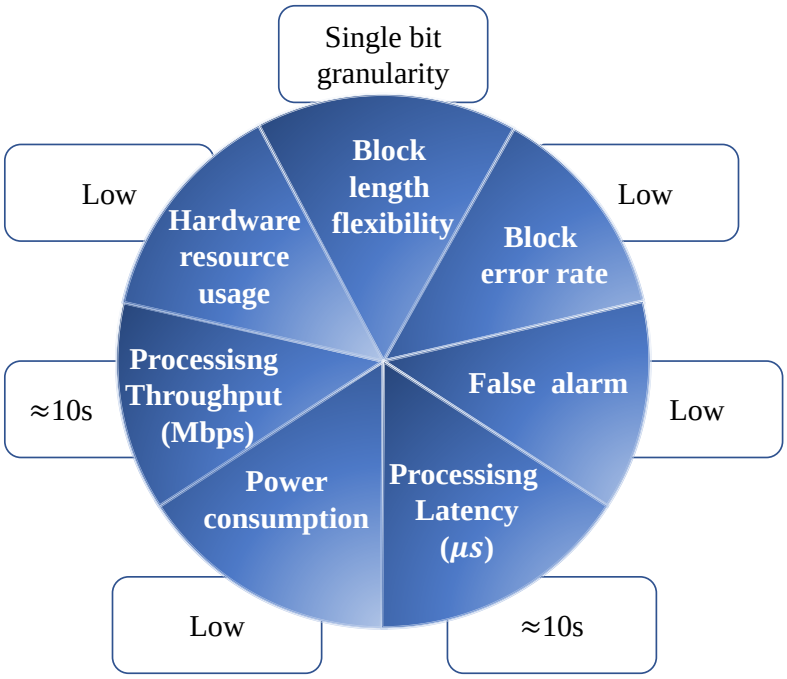


Figure 1.1 – Design criteria of the NR polar codes.

rate flexibilities render unsuitable most of the already published hardware polar decoder implementations at that time. This motivates our thesis work in proposing an original flexible hardware architecture for decoding 5G NR polar codes that can support all the frame sizes and code rates defined in 3GPP with a low decoding latency. The approach used as part of this work is to first conduct a comprehensive study on polar codes and their decoding algorithms, then to carry out a thorough design space exploration of polar decoder architectures to finally propose a novel efficient hardware architecture targeting FPGA devices. Motivated by the importance of blind decoding in 5G NR control channel, we investigate decoding multiple frames aiming at supporting efficiently this feature as well.

Contributions

The main contributions of this PhD thesis work can be summarized as follows:

Design space exploration of polar decoder architectures for 5G NR polar codes: The stringent requirements introduced by the 5G standard in terms of block length and code rate flexibility, together with low end-to-end latency and high error correction performance

represent a major challenge for their hardware implementation. Therefore, a thorough design space exploration of polar decoder architectures has been conducted with the following main contributions:

— The development of a complete software simulation environment of polar coding/decoding for performance evaluation under different algorithms in both floating point and fixed point data representation.

— The study of the impact of main code and decoder design parameters on the latency, throughput, and the hardware complexity of semi-parallel decoding architectures. The impact of these parameters on the hardware efficiency of semi-parallel architectures is significant.

— The proposal of two multi-frame decoding approaches that increase the throughput and improve the utilisation rate of the processing units of these architectures.

— The elaboration of detailed analytical and logic synthesis results for a large range of parameters values in order to constitute a reference for the implementation of flexible, yet efficient FEC decoders for polar codes.

Novel low latency efficient decoder design for 5G NR polar codes:
Based on the design space exploration study mentioned above, and motivated by the need to provide a hardware-efficient polar decoder that supports the required flexibility and latency levels for 5G NR, we proposed a novel efficient hardware architecture offering the following features:

— Downlink and uplink 5G NR control channel compliance with full rate and frame size support, ranging from $N = 32$ to $N = 1024$ bits.

— CRC-aided $L = 8$ SCL decoder with a semi-parallel architecture [55] for best performance.

— Dedicated specific constituent-code decoders for reduced latency.

— Measured FPGA-based worst-case decoding latency of 23.91 μs, in compliance with target 5G NR constraints.

— Favourable comparison with previously-published designs.

— Support of multi-frame decoding.

The above-mentioned features were obtained thanks to the following hardware architecture-related contributions:

— The proposal and implementation of on-the-fly identifier for the number of constituent codes in addition to their type and length.

— The introduction of an original special node decoder capable of decoding all identified constituent-code types.

— The development of hardware-efficient partial-sum (PS) and CRC modules supporting multi-bit decoding.

List of publications

The results of this thesis work have led to the following publications:

Conference papers

1. O. Mouhoubi, C. Abdel Nour and A. Baghdadi, "On the Latency and Complexity of Semi-Parallel Decoding Architectures for 5G NR Polar Codes," *International Symposium on Signal, Image, Video and Communications* (ISIVC), 2022.
   This conference paper presents the first results on the design space exploration of semi-parallel hardware architectures for decoding 5G NR polar codes. The impact of main code and decoder design parameters on the latency and the hardware complexity are studied and analysed.
2. O. Mouhoubi, C. Abdel Nour and A. Baghdadi, "Low Latency Architecture Design for Decoding 5G NR Polar Codes," *International Workshop on Design and Architectures for Signal and Image Processing* (DASIP), 2022.
   This conference paper proposes an original special node list decoder and a special node identifier which help the proposed decoder to continue to benefit from tree pruning techniques to speed-up the decoding whilst maintaining compliance with the 5G NR and the various defined combinations of code rate and code length. The first part of Chapter 4 discusses the contributions of this paper.

Journal papers

1. O. Mouhoubi, C. Abdel Nour and A. Baghdadi, "Latency and Complexity Analysis of Flexible Semi-Parallel Decoding Architectures for 5G NR Polar Codes," *IEEE Access*, 2022, doi: 10.1109/ACCESS.2022.3216292. **(Published).**
   This journal paper presents a detailed design space exploration of semi-parallel

hardware architectures that we conducted and that can be used to design efficient decoders for 5G NR polar codes. It extends the ISIVC'2022 conference paper with results on throughput, hardware efficiency, activity and utilisation rate of the hardware resources, and the proposal of two multi-frame decoding approaches to increase the throughput and improve the utilisation rate of the decoder processing units. Chapter 3 discuss the contributions of this paper.

2. O. Mouhoubi, C. Abdel Nour and A. Baghdadi, "Hardware Design and FPGA Implementation of Low Latency Decoder for 5G NR Polar Codes," *IEEE Access* **(Submitted).**
   This journal paper proposes an original hardware architecture for decoding the 5G NR polar codes of the uplink and the downlink control information channel. Results of this paper are discussed and compared to the state-of-the-art FPGA polar deocder implementations in Chapter 4.

Manuscript organization

The rest of this manuscript is organized in three chapters as follows:

Chapter 2 reviews the basic concepts related to polar codes including their encoding and decoding process. After introducing the principle of polarization, numerous approaches for decoding polar codes with various levels of complexity, throughput, latency and efficiency are presented. The low-complexity successive cancellation decoder and its list-based extension are detailed, along with their simplified variants based on tree-pruning techniques. Since our interest is focused on 5G NR polar codes, a detailed presentation of their encoding and decoding process is provided.

Chapter 3 is dedicated to the presentation of the proposed design space exploration of polar decoder architectures for 5G NR polar codes. The chapter starts by describing the developed software simulation environment of polar coding/decoding for performance evaluation under different algorithms in both floating point and fixed point data representation. Then, the study of the impact of main code and decoder design parameters on the latency, throughput, and the hardware complexity of semi-parallel decoding architectures is provided. The impact of these parameters on the hardware efficiency of semi-parallel architectures is significant. Therefore, two multi-frame decoding approaches are proposed to increase the throughput and

improve the utilisation rate of the processing units of these architectures. Detailed analytical and logic synthesis results are provided and compared for a large range of values in order to constitute a reference for the implementation of flexible, yet efficient FEC decoders for polar codes.

Chapter 4 presents our contributions related to the design of a novel hardware-efficient polar decoder that supports the required flexibility and latency levels for 5G NR. While offering multiple features in compliance with target 5G NR constraints including a full rate and frame size handling of downlink and uplink control channel, this novel hardware architecture targeting FPGA devices compares favourably with previously-published designs. Furthermore, an efficient multi-frame decoding scheme particularly suited for blind decoding of downlink control information is proposed with an efficient implementation.

Chapter 2

Polar codes

Contents

This chapter introduces the basic concepts related to polar codes including their encoding and decoding process with an emphasis on the polar codes adopted in 5G NR. It starts by introducing the founding concept of channel polarization. Then, the successive cancellation decoding algorithm and its list-based extension are described followed by a brief review on other forms of decoding polar codes. Afterwards, simplified successive cancellation decoding variants applying what is known by tree-pruning are presented as means to significantly reduce the latency and increase the throughput. Finally, an overview of the polar codes adopted in 5G NR is provided.

2.1 Polar codes

Proposed in the last few years, polar codes [9] represent one of the latest additions to the family of forward error correction (FEC) codes. When the codeword length N tends to infinity, they have been shown to achieve channel capacity in binary discrete memoryless channels under the low complexity successive-cancellation (SC) algorithm [10]. However, their performance starts to degrade at practical code lengths. Thanks to their low complexity encoding and decoding under SC algorithm that have attracted the attention of academia and industry in the past decade, they have been recently adopted in the New Radio (NR) 5G standard for uplink and downlink control channels where polar codes with short to moderate block lengths are specified [3]. LDPC code family [35, 71] is meanwhile adopted for the data channel of the enhanced mobile broadband (eMBB) service.

2.1.1 Channel polarization

The particularity of polar codes is linked to the polarization phenomenon. In order to explain this, we consider a binary discrete memoryless channel (B-DMC) $\mathcal{W}$ with input and output alphabet $\mathcal{X}$ and $\mathcal{Y}$, respectively. Moreover, transition probability $\mathcal{W}(y|x)$, where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. We can now define the symmetric capacity $I(\mathcal{W})$ as the highest rate at which reliable communication can occur over $\mathcal{W}$. Hence, when $I(\mathcal{W})$ tends to 1, it indicates error-free transmission; however, when it tends to 0, it indicates that information transmission is impossible. Furthermore, we define the Bhattacharyya parameter $Z(\mathcal{W})$ as a measure that provides an upper limit on the probability of erroneous detection when Maximum Likelihood (ML) is used to estimate a single received symbol y. These two

metrics are expressed as a function of $\mathcal{W}(y|x)$ as follows:

$$I(\mathcal{W}) = \sum_{y\in\mathcal{Y}} \sum_{x\in\mathcal{X}} \frac{1}{2}\mathcal{W}(y|x) \log \frac{\mathcal{W}(y|x)}{\frac{1}{2}\mathcal{W}(y|0) + \frac{1}{2}\mathcal{W}(y|1)} \tag{2.1}$$

and

$$Z(\mathcal{W}) = \sum_{y\in\mathcal{Y}} \sqrt{\mathcal{W}(y|0)\mathcal{W}(y|1)}. \tag{2.2}$$

Both metrics have values between 0 and 1 and are related by these two inequalities on the B-DMC channel:

$$I(\mathcal{W}) \geq \log \frac{2}{1 + Z(\mathcal{W})}, \tag{2.3}$$

$$I(\mathcal{W}) \leq \sqrt{1 - Z(\mathcal{W})^2}, \tag{2.4}$$

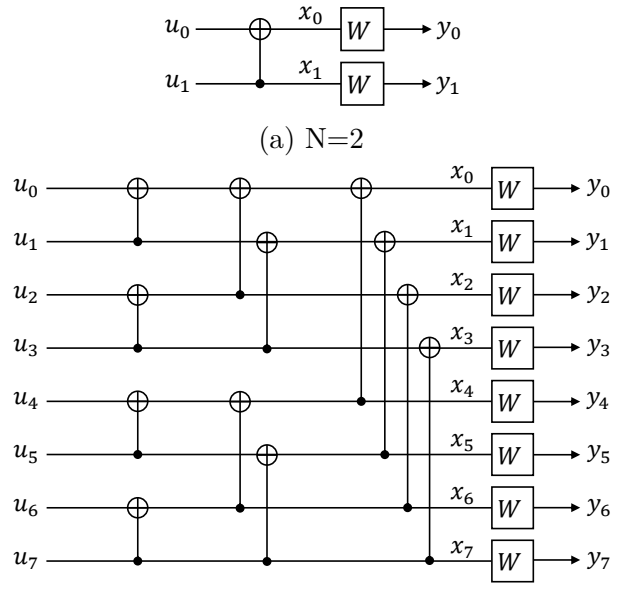
from which it can be observed that $I(\mathcal{W})$ and $Z(\mathcal{W})$ are inversely proportional. The group consisting of polar encoder/decoder and channel can be seen as a set of N channels where each channel transmits one bit. The mutual information between the information $(u_0, u_1) \in \mathcal{X}^2$ and received values $y_0, y_1 \in \mathcal{Y}^2$, when the transmission is carried out using two independent instances of $\mathcal{W}$, is given by :

$$I(Y_0, Y_1; U_0) = I(\mathcal{W}) = I(Y_0, Y_1; U_1). \tag{2.5}$$

The *polarization* term conveys that the N channels are split into two groups. A group of very reliable channels with a very low probability of error, and a group of unreliable channels, with a high probability of error. Equivalently, if u_0, u_1 are transformed into $(x0, x1)$ as shown if Fig. 2.1a so that $x_0 = u_0 \oplus u_1$ and $x_1 = u_1$, the mutual information values between the information and received symbols become:

$$I(Y_0, Y_1; U_0) \leq I(\mathcal{W}) \leq I(Y_0, Y_1; U_1)\ . \tag{2.6}$$

This means that the probability of correctly estimating u_1 increases while correctly estimating u_0 decreases. Fig. 2.1b shows the polarization transformation for $N = 8$. Proof of this inequality is presented in [9]. It is now possible to show that the proportion of reliable channels for a given code rate R asymptotically achieve the channel capacity as defined in information theory [81] when the number of transformations increases and code length N tends to infinity.



(a) N=2

(b) N=8

Figure 2.1 – Polarization transformation for polar codes of length $N = 2$ and $N = 8$.

2.1.2 Polar coding

Polar codes apply the channel polarization transform that divides the bit-channels into either perfect or completely noisy channels. Then they allocate information bits to the K most reliable bit-channels while the remaining bits are frozen, i.e., are all set to a known value, usually '0'. Equivalently, for a codeword length $N = 2^n$, $n \geq 1$, a (N, K) polar code is a block code with K input bits and N output bits whose generator matrix G is the $n-$th Kronecker power of matrix $F = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$, i.e., $G_N = F^{\otimes n} = \begin{bmatrix} F^{\otimes n-1} & 0_{n-1} \\ F^{\otimes n-1} & F^{\otimes n-1} \end{bmatrix}$. The encoding process is performed by matrix multiplication $x = u.G$, where $u = (u_0, u_1, \ldots, u_{N-1})$ stands for the sequence input vector consisting of information bits and frozen bits, and $x = (x_0, x_1, \ldots, x_{N-1})$ stands for the encoded vector.

Example 2.1.1 *for the polar code PC(8,4) and* $u = [0, 0, 0, u_3, 0, u_5, u_6, u_7]$ *the corresponding codeword is:*

$$[0,0,0,u_3,0,u_5,u_6,u_7]\times\begin{bmatrix}1&0&0&0&0&0&0&0\\1&1&0&0&0&0&0&0\\1&0&1&0&0&0&0&0\\1&1&1&1&0&0&0&0\\1&0&0&0&1&0&0&0\\1&1&0&0&1&1&0&0\\1&0&1&0&1&0&1&0\\1&1&1&1&1&1&1&1\end{bmatrix}=\begin{bmatrix}u_3+u_5+u_6+u_7\\u_3+u_5+u_7\\u_3+u_6+u_7\\u_3+u_7\\u_5+u_6+u_7\\u_5+u_7\\u_6+u_7\\u_7\end{bmatrix}$$

The codeword obtained by the multiplication of U and $F^{\otimes n}$, as illustrated in the Example 2.1.1, is non-systematic since information bits are not members of the codeword. A forward error correction code is systematic if information and parity bits can be clearly distinguished. Systematic codes have the advantage that the parity data can simply be appended to the source block, and receivers do not need to recover the original source symbols if received correctly.

2.2 Decoding algorithms of polar codes

Typically, a decoding algorithm is a process in which the received information at the output of the channel is treated to retrieve the transmitted information with the least amount of possible errors. Decoding algorithms are more or less complex, fast, and efficient. To date, numerous approaches for efficient decoding of polar codes have been reported since the first decoding algorithm for polar codes has been proposed in [9], successive cancellation decoder (SC). Although SC yields very good performance for long polar codes, it is significantly degraded with respect to Maximum Likelihood decoding performance at short to moderate block length. To overcome this issue, Successive Cancellation List (SCL) [87], Flip Successive Cancellation (SCF) [4], and Stack Successive Cancellation (SCS) [73], all derived from the SC decoders, have been proposed to the detriment of extra complexity. However, due to the serial processing nature of the SC-based decoding algorithm, all the algorithms mentioned above have a high decoding latency and low throughput, which has a significant impact on their practical applications. While these algorithms are characterized by having hard outputs (i.e., output are bits), fully parallel decoding algorithm, such as belief propagation (BP) with soft output have drawn lots of attention. The performance of the BP decoding was studied in [11, 51] based on Forney's factor graph representation

[8]. Results demonstrated that BP decoding outperform SC decoding in terms of decoding latency and throughput.

2.2.1 Successive Cancellation decoding algorithm

Suitable for hardware design, the low complexity successive-cancellation (SC) algorithm is one of the most common decoding techniques proposed for polar codes [9]. Although sufficient for long polar codes, its error correction performance degrades significantly for medium and short code lengths. We denote the source vector as u_i^N, consisting of information and frozen bits. As the name suggests, the SC decoding algorithm estimates u_0, then u_1 and so on until u_{n-1} sequentially by observing the channel output y^N. The estimate $\hat{u}_i$ is obtained based on all the previous estimates $\hat{u}_0$ to $\hat{u}_{i-1}$, denoted by $\hat{u}^{i-1}$, according to the following rule:

$$\hat{u}_i = \begin{cases} 0 & \text{if } \frac{\Pr\left(\mathbf{y}|\hat{u}^{i-1}, u_i=0\right)}{\Pr(\mathbf{y}|\hat{u}^{i-1}, u_i=1)} > 1, \\ 1 & \text{otherwise.} \end{cases} \tag{2.7}$$

The decoding flow of the SC decoder follows the butterfly structure (bipartite graphs) of Fig. 2.2, illustrated for a polar code of length $N = 8$ bits. The graph consists of $\log_2 N - 1$ stages, where each stage is made up of N nodes. The decoder soft-inputs denoted by $L_{n,i}$, where $(0 \leq i \leq N-1)$ is the index of the graph row, are provided at the right side of Fig. 2.2. These inputs are processed in several steps during decoding while proceeding towards the left side by applying functions f and g drawn in white and grey colors, respectively. Each node f has two input LLRs denoted by $L_{j,i}$ and $L_{j,i+2^j}$ and each node g has two input LLRs denoted by $L_{j,i-2^j}$ and $L_{j,i}$, where j $(0 \leq j \leq log_2 N - 1)$ is the index of the decoding stage. A hardware-friendly version of soft value updating is carried out in log-likelihood ratio (LLR) domain. The f function applies the Min-Sum approximation as follows:

$$f(L_{j+1,i}, L_{j+1,i+2^j}) = \text{sign}(L_{j+1,i}.L_{j+1,i+2^j}).\min(|L_{j+1,i}|, |L_{j+1,i+2^j}|). \tag{2.8}$$

The bit-estimates are provided sequentially from top to bottom at the left side of the butterfly structure. To do so, i.e. to apply the decoding steps, these estimates are combined and fed-back to the decoder. Denoted by $\hat{s}_{i,j}$, the combination of the previously decoded codeword bits results into a *partial sum* at node i within the decoding stage j. These partial sums are also computed sequentially based on the already-decoded source bits as

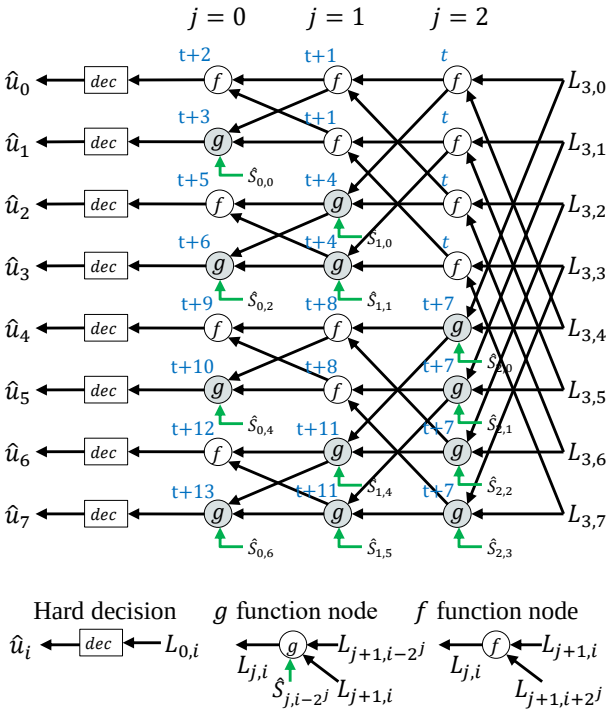


Figure 2.2 – Butterfly representation of the SC decoder for $N = 8$.

illustrated by Figure 2.3. Furthermore, they are required for the computation of the g function according to the following rule:

$$g(L_{j+1,i-2^j}, L_{j+1,i}) = (1 - 2\hat{s}_{j,i-2^j}).L_{j,i-2^j} + L_{j,i} \ . \tag{2.9}$$

In the example of Fig. 2.2, $\hat{s}_{1,1} = \hat{u}_1$ is the partial-sum needed to compute the g function of the first visited node of stage $j = 1$ while $\hat{s}_{1,4} = \hat{u}_4 \otimes \hat{u}_5$ is the partial sum needed to compute the g function of the third visited node of the same stage. At every decoding stage j of the butterfly structure, 2^j LLRs are used and half of them are produced for the next lower stage $j - 1$. Moreover, each stage j is activated $\frac{N}{2^j}$ times during the decoding of a frame of length N. Assuming that the processing of an activated stage can be performed in one clock cycle, the total number of clock cycles required to decode one frame is:

$$\mathbb{L}_{ref} = \sum_{j=0}^{n-1} 2^{n-j} = 2N - 2. \tag{2.10}$$

This corresponds to $\mathbb{L}_{ref} = 14$ in the example of Fig. 2.2 where the followed schedule for node activation is indicated by the label (t+l_{ref}) with $0 \leq l_{ref} < \mathbb{L}_{ref}$. Considering the case

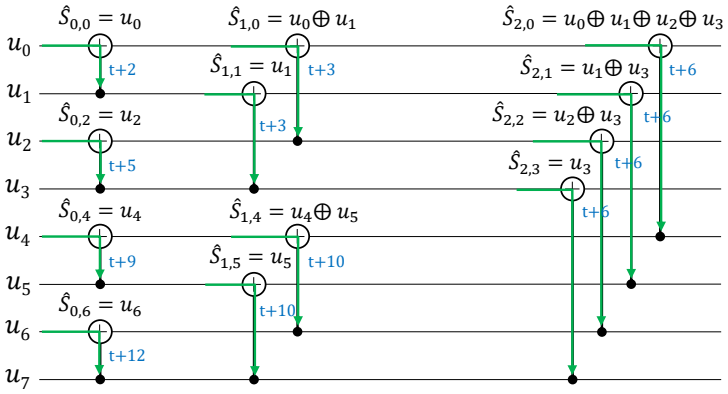


Figure 2.3 – Factor graph of partial sums computation.

of a practical SC decoding process of a polar code of length $N = 8$, synchronous sequential logic analysis is illustrated in Fig. 2.4. The rising edges of the clock corresponding to the different time periods of the decoding steps labeled at the top of each node of Fig. 2.2. The first f operation starts at clock cycle t, which corresponds to the activation of stage $j = 2$. The produced LLRs are stored in a dedicated memory and used at the second rising edge of the clock to perform f functions of stage $j = 1$. When the clock cycle $t + 2$ arrives, the first node f of stage $j = 0$ is updated, and the first estimate $\hat{u}_0$ is obtained through hard decision. In the meantime, $\hat{u}_0$ is fed into the decoding circuit as partial sum bit $\hat{s}_{0,0}$. They are used together with the $L_{1,0}$ and $L_{1,1}$, calculated at $t + 1$ in order to update the first g node of stage $j = 0$ at $t + 3$. In this way, $\hat{u}_1$ is estimated and partial sums $\hat{s}_{1,0} = \hat{u}_1$, $\hat{s}_{1,1} = \hat{u}_0 \otimes \hat{u}_1$ are produced. This enables the computation of $L_{1,2}$ and $L_{1,3}$ by the two first g nodes of stage $j = 1$ at the rising edge of clock cycle $t + 4$. The remaining bits are estimated in the same way during the next clock cycles until the estimation of $\hat{u}_7$ at the rising edge of clock cycle $t + 13$.

2.2.2 List-SC decoding algorithms

The major drawback of the SC algorithm resides in its inability to recover from wrong bit estimates, especially at the early stages of decoding. This leads to erroneous partial sum computations and potential error propagation. Based on this observation, a SCL algorithm was proposed in [87] to avoid resorting to hard decisions when computing partial sums during the sequential decoding phase. Indeed, hard decisions were replaced by soft hypotheses for the error-prone bits identified by low reliability values. This leads to the simultaneous exploration of several codeword candidates or equivalently paths in the graph

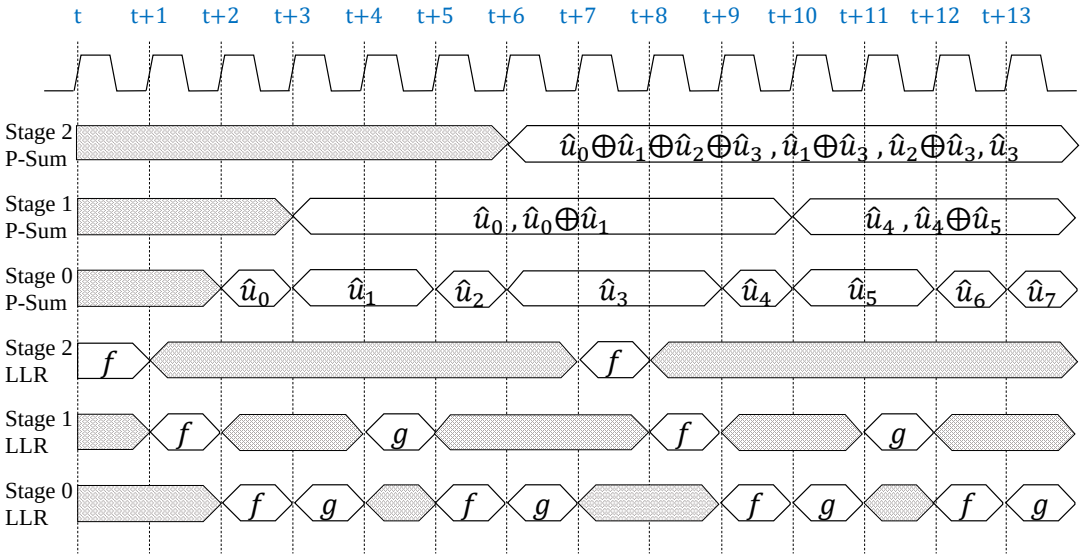


Figure 2.4 – Synchronous sequential logic of decoding polar code of length $N = 8$.

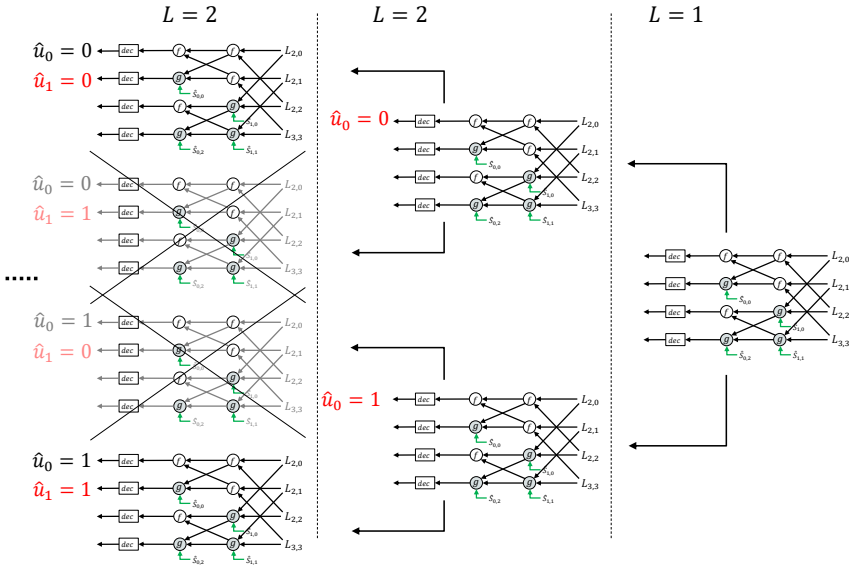


Figure 2.5 – List SC decoding of polar code (4,4) with $L = 2$

of Fig.2.2, each corresponding to one or more varying bit-hypotheses. Hence, for each bit u_i decoding step, both its possible values 0 and 1 are considered and $2L$ new candidate paths are explored. It should be noted that when the encountered bit is frozen, path duplication is not applied since the value of the frozen bits is unchanged. However, in order to break the exponential growth in the number of candidate paths, a subset L of the most likely paths is set to survive as illustrated in figure 2.5. The choice is made by selecting the L lowest [15, 18, 52, 54, 58, 91] path metric (PM) values computed as follows:

$$\begin{aligned} &\mathrm{PM}_{-1_l} = 0, \\ &\mathrm{PM}_{i_l} = \begin{cases} \mathrm{PM}_{i-1_l} + |L^l_{0,i}|, & \text{if } \hat{u}_{i_l} \neq \frac{1}{2}\left(1 - \mathrm{sgn}\left(L^l_{0,i}\right)\right), \\ \mathrm{PM}_{i-1_l}, & \text{otherwise.} \end{cases} \end{aligned} \tag{2.11}$$

In terms of complexity, the SCL decoder can be seen as the concatenation of L competing SC decoders. Assuming that a path selection can be performed in one clock cycle, the latency of the SCL decoder with K information bits becomes:

$$\mathbb{L}_{\mathrm{SCL}}(N, K) = \mathbb{L}_{ref} + K = 2N + K - 2. \tag{2.12}$$

The operation of path duplication involved in the SCL algorithm occurs $\mathcal{O}(LN)$ times and requires the copy of a data structure of at least $\mathcal{O}(N)$. The complexity of a straightforward implementation is at least $\mathcal{O}(LN^2)$. Nevertheless, authors in [86] propose a way to implement the SCL decoder with time complexity $\mathcal{O}(LN \log N)$ instead of $\mathcal{O}(LN^2)$. by the use of the *copy-lazy* technique. Simulation results in [86] show that for a (2048, 1024) polar code, a list size of $L = 32$ is able to achieve an error correction performance close to the ML decoder.

In [87], authors observe that in the majority of cases where list decoding fails to find a correct codeword, the latter appears among the $L - 1$ remaining paths that have been declassified by the decoder since they have been considered to be less likely. Therefore, they conclude that if the decoder can be assisted in its final choice, it would improve significantly the performance of polar codes. The best candidate error-detecting code to carry out this task is the cyclic redundancy check (CRC) code which is widely used in practical communication systems. To do so, a L_{CRC} bits are added to the underlying polar code to form $K + L_{\mathrm{CRC}}$ non-frozen bits while the effective code rate of the code is still unchanged. A combination of the SCL decoder with a CRC detector was proposed in [72]

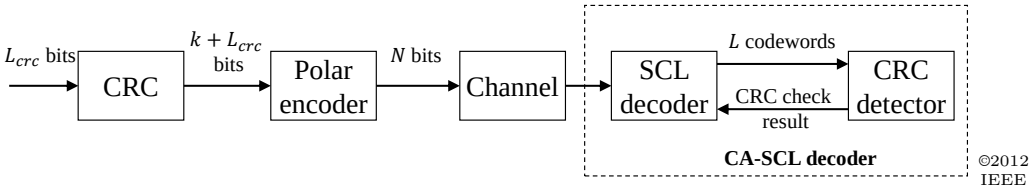


Figure 2.6 – Encoding and decoding schemes of CA-SCL decoder [72].

and [57]. The corresponding encoding and decoding scheme is depicted in figure 2.6. At the receiver side, when the decoding process is completed, the SCL decoder outputs the final L most likely codewords to the CRC detector and the most likely one that passes the CRC check would be selected as the output codeword. Such a decoding scheme is referred to as CRC-aided SCL (CA-SCL).

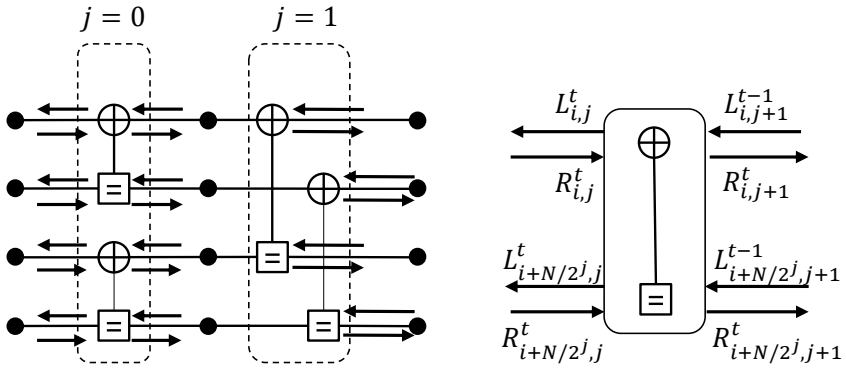
While the CA-SCL algorithm significantly improves the decoding performance of the SC algorithm, it increases its computational complexity. In [57], a new algorithm is proposed where both SC and CA-SCL algorithms are combined to create an adaptive variant that can benefit simultaneously from the error correction capability of the former and approach the low complexity of the latter. starting from a size 2-list decoder, the Partially Adaptive SCL (PASCL) algorithm of [57] increases gradually the list size every time the current paths fail the CRC check operation until reaching the maximum list size L_{max}. In a another approach, the Fully Adaptive SCL (FASCL) algorithm of [78], starts as SC until a CRC check is negative, the decoder is switched to SCL going directly from one-list size to $L_{\max}$-list size decoding. The downside of these type of algorithm is the high decoding latency which is equal to the sum of the latency of decoding SC and the intermediate SCL decoders.

An evaluation of polar code decoders, in particular, SCL, SSCL, Fast-SSCL, and Partitioned SCL (PSCL) [48] decoders, is provided in [29]. The implementation results of these decoders have shown that polar decoders have reduced area, power and energy consumption and a comparable error-correction performance with comparison to the WiMAX LDPC architectures [50, 64, 65], which make them suitable for 5G communications.

2.2.3 Iterative decoders

The Belief propagation (BP) decoder

Unlike SC-based algorithms, the Belief Propagation (BP) algorithm [11, 101] provides soft decision values at the end of the decoding process instead of hard decision bits. Thus,



(a) BP factor graph for $N = 4$. (b) BP processing unit

Figure 2.7 – Factor graph and processing unit of BP decoder.

the output of this algorithm can be used as input vector to other soft-input decoders. The BP algorithm was first proposed by Arikan [10]. It consists of passing LLRs iteratively through the nodes of the factor graph which are used for updating propagated messages. It allows the messages to be exchanged in both directions from left to right and from right to left. The decoding process of the BP algorithm over the factor graph is illustrated in figure 2.7.a for $N = 4$ where each node is associated with two messages $L^t_{i,j}$ and $R^t_{i,j}$ denoting the left-to-right and the right-to-left likelihood messages of the i^{th} node at the j^{th} stage and the t^{th} iteration, respectively. These messages are propagated between adjacent nodes and updated by a processing element using the min-sum approximation according to the following equations:

$$
\begin{aligned}
L^t_{i,j} &= sign(L^{t-1}_{i,j+1}) \cdot sign(L^{t-1}_{i+N/2^j,j+1} + R^t_{i+N/2^j,j}) \cdot \min(|L^{t-1}_{i,j+1}|, |R^t_{i+N/2^j,j}|).\\
L^t_{i+N/2^j,j} &= L^{t-1}_{i+N/2^j,j+1} + sign(L^{t-1}_{i,j+1}) \cdot sign(R^t_{i,j}) \cdot \min(|L^{t-1}_{i,j+1}|, |R^t_{i,j}|).\\
R^t_{i,j+1} &= sign(R^t_{i,j}) \cdot sign(L^{t-1}_{i+N/2^j,j+1} + R^t_{i+N/2^j,j}) \cdot \min(|R^t_{i,j}|, |L^{t-1}_{i+N/2^j,j+1} + R^t_{i+N/2^j,j}|).\\
R^t_{i+N/2^j,j+1} &= R^t_{i+N/2^j,j} + sign(L^{t-1}_{i,j+1}) \cdot sign(R^t_{i,j}) \cdot \min(|L^{t-1}_{i,j+1}|, |R^t_{i,j}|)
\end{aligned}
\tag{2.13}
$$

During an iteration the $N/2$ processing units implemented by the BP decoder are activated at each stage in the left-right direction. When the number of iterations I reaches $I_{\max}$, the decoder outputs the estimated information bits based on the hard decision of the right-most messages $R^{I_{\max}}_{i,\log_2 N+1}$. The processing unit operations are illustrated in Fig. 2.7.b

The Soft CAncellatioN (SCAN) decoder

Similarly to the BP decoder, the Soft Cancellation (SCAN) decoder proposed in [33] is a soft output iterative message passing algorithm. The decoding schedule of the SCAN decoder is similar to the one of the SC algorithm. However, the real LLR value which corresponds to the bit estimate is retained instead of applying a hard decision. Afterwards, the LLR value is used to update the LLRs exchanged in the left to right direction on the decoder factor graph. After reaching a maximum number of iterations defined beforehand, SCAN decoding applies hard decisions on left-hand side LLRs corresponding to estimates of the $\hat{\boldsymbol{u}}$ vector. When compared to the BP decoder, SCAN converges faster and requires less iterations since it relies on the natural SC scheduling. For instance, it only takes two iterations to SCAN in order to outperform the SC and the BP (with 40 iterations) when decoding a (256, 128) polar code [76].

2.2.4 Other forms of decoding polar codes

The SC flip decoder

The SC flip decoder originally proposed in [4] is an alternative way of decoding polar codes under the SC algorithm. A CRC detector is associated to the SC decoder in order to improve its error-correction capability at the cost of a resulting variable decoding latency. However if the CRC verification indicates an erroneous codeword at the end, the decoding process is restarted by flipping the decision made on the least reliable LLR. The CRC check is performed at the end. This process is repeated for a predefined number of times specified beforehand. Several studies looked into improving SC flip decoding by expanding it to list SC flip decoding and then proposing the dynamic, partitioned and BP SC flip decoding [19, 20, 25, 30, 31, 82, 97, 98, 103].

The SC Stack decoder

The SC stack algorithm [73, 84] is a variant of the SCL algorithm, which as the name implies, uses a stack of a certain depth to pile up the path metrics associated to the different L candidate paths. Indeed, while the SCL performs candidate competition of the L paths simultaneously, the SC stack extends candidate paths with the ML in the stack, and therefore paths in the stack no longer keep the same length. When a leaf node is reached, the decoder extends the most probable path from the stack. A CRC check is

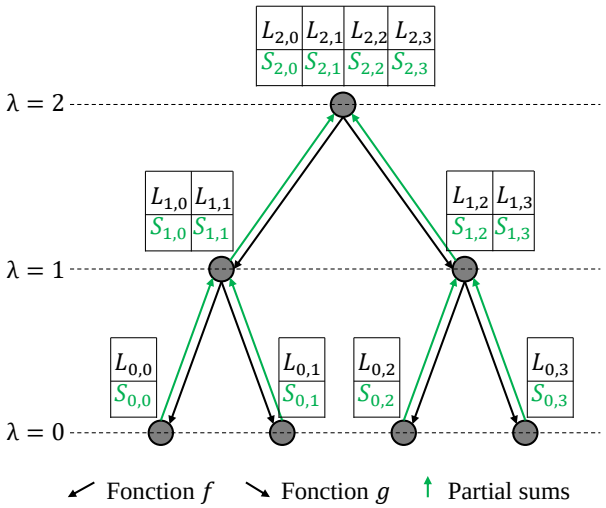


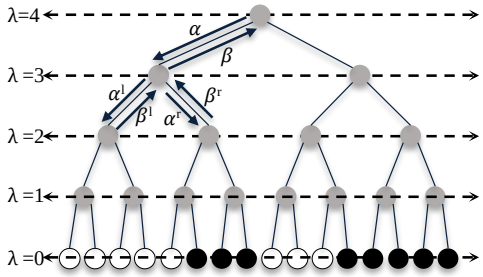
Figure 2.8 – Tree representation of SC decoding for $N = 4$.

performed after decoding the last bit and the decoding ends if a CRC check is successful or until a maximum of tested path candidates is reached [94].

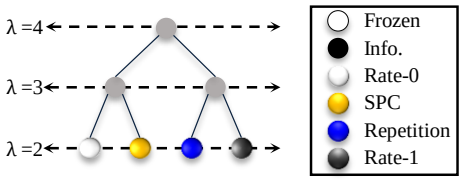
2.3 Tree-pruning techniques of polar codes

2.3.1 Tree representation of SC polar decoders

The butterfly structure of Fig. 2.2 is not the only way to represent polar codes. Indeed, a binary tree is another natural way to do so. Due to its recursive construction, a polar code of length N can be represented as the concatenation of two polar codes of length $N/2$. The tree representation of SC decoding of polar code of length $N = 4$ is shown in Fig. 2.8. The two types of data that are processed during the decoding, in this case, LLRs and partial sums are arranged on $\log_2 N + 1$ stages. Each stage λ, with $(0 \leq \lambda \leq \log_2 N)$, comprises $\frac{N}{2^\lambda}$ node. The root node $(\lambda = \log_2 N)$ includes the channel LLRs and the final partials sums. in the case of decoding a non systematic polar code, the root node includes the decoded sequence $\hat{u}^{n-1}$ instead of final partial sums. Bit estimation is performed at leaf nodes, i.e., $\lambda = 0$. In the intermediate tree-stages $(0 < \lambda < \log_2 N)$, N LLRs and N partial sums are calculated in $\frac{N}{2^\lambda}$ nodes at different period times. The nodes compute 2^λ and LLRs produce $2^{\lambda-1}$ LLRs. The decoding process starts from the root node where all the N available LLR values are computed to produce $N/2$ LLRs for left node of the foremost adjacent decoding stage. This latter uses its LLRs to produce, in turn, $N/4$



(a) SC decoder tree



(b) SC decoder pruned tree

Figure 2.9 – SC based decoder tree and its corresponding pruned tree of PC(16,8).

LLRs for the left node to which it is connected at the next lower decoding stage and so forth up until reaching the leaf node with a single LLR value. This tree representation of the decoder highlights better the parallelism degree of the decoder which is progressively reduced from N to 1 from root node to leaf nodes.

2.3.2 SC and SCL tree-pruning techniques

A simplified version of the SC decoding algorithm (SSC) was presented in [5] in which the SC decoding tree is pruned. In fact, a tree with only frozen bits at leaves does not need to be traversed since its output is already known and is equal to an all-zero vector. This type of node is referred to as R0. Moreover a tree with only information bits can directly be decoded by applying a threshold decision on the tree root. This type of node is referred to as R1. Furthermore, the authors in [79] have identified two new types of nodes among the constituent codes of rate $0 < R < 1$. Hence, a repetition node (REP) is a constituent code where all the bits are frozen except for the last one. The partial sum values of a Rep node are either all equal to 0 or all equal to one. All of the LLR values of the considered node are added together to decide which values the partial sums should take. If the total of the summation is greater than 0, all partial sums are set to 0, otherwise they are all set to 1. Furthermore, the single parity check (SPC) node is a constituent code where at

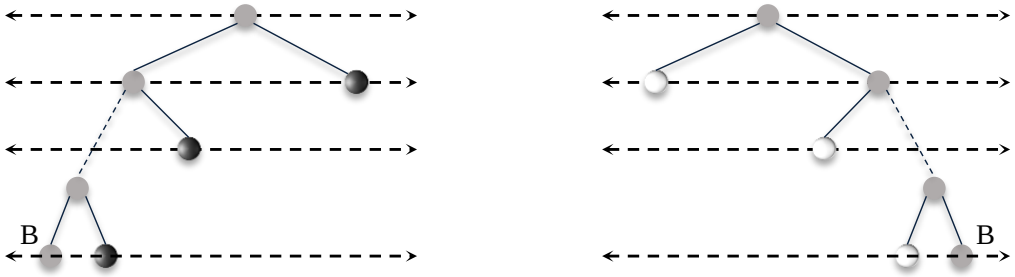
the exception of the first bit, all the bits are information. The decoding of this type of nodes requires multiple steps. First, hard decision is applied to each LLR value similarly to the decoding of R1 nodes. The parity of the obtained partial sums is then checked. If the parity check yields a value of 0, partial sums are retained and decoding of the SPC node is complete, if not, the partial sum associated with the lowest absolute LLR value is inverted beforehand. The original SC decoding tree of PC(16,8) and the pruned SC decoder tree of the Fast-SSC decoder are illustrated in Fig.2.9 where the four types of constituent codes are colored differently.

The pruning mechanism can also be applied on a SCL [45] and a SCF [26, 36, 102] decoders. However, the decoding of the pruned sub-trees is different from the case of SC. Indeed, in SCL decoding, the number of paths is duplicated when meeting an information bit. Except for R0 node whose leaves are frozen, the remaining specific constituent codes that feature at least one information bit need to perform one or multiple duplications as part of their decoding process. The decoding steps of these nodes in SCL decoding are detailed as follows:

— R0 node: Although it is not required to perform path duplication, the PM needs to be updated as was the case with SCL decoder. Each LLR value related to the top R0 node is compared to the 0-value. The PM is then penalized with the absolute value of the LLR if its LLR value is negative.

— Rep node: it includes only one information bit. Thus, the paths are duplicated one time considering both possibilities corresponding to all-zero and all-one partial sum vectors. The PM of each candidate is computed by considering both possible hypothesis, i.e., 0 and 1, on the single information bit. This is done by summing up the absolute value of all the negative LLRs on one side and summing up the absolute value of all the positive LLRs on the other side.

— R1 node: All the bits of this node are information, the number of path duplications required to decode such a node is equal to the length of the node. For each LLR value, a partial sum is estimated and both its possible values 0 and 1 are considered. The two candidate paths generated for these values update the PM following the sign of the LLR. Indeed, the paths that take the 0-valued partial sum maintain the value of the PM if the LLR is positive as well. Moreover, the paths that take the 1-valued partial sum maintain the value of the PM if the LLR is negative. Otherwise, the absolute value of the LLR is added to the PM of the path. When all the PMs are split, they are sorted and the paths identified having lowest PMs are kept to



(a) Type-III and Type-IV decoding tree (b) Type-I, Type-II and Type-V decoding tree

Figure 2.10 – Decoding tree of new special nodes Type-I to Type-V.

pursue the decoding. This path split process is applied to all the LLRs of the node.

— SPC node: It is similar to the R1 node in the way that it needs multiple path forks to decode information bits. However, it comprises a single frozen bit which is decoded first. Using the LLR values at the top of the sub-tree node, the least reliable bit corresponding to the minimum absolute value is found and the parity equation is tested. If the latter is not satisfied the PMs are initialized by adding the absolute value of the least reliable bit. In the second step, the remaining bits are decoded the same way R1 bits are decoded applying as many time path forks as the number of information bits. However, PM update is slightly different. In this case, the minimum LLR value is either deduced from the PM or added to it depending on the parity check result, i.e if the partial sums and the sign of LLRs do not match. The final step consists of satisfying the parity check, the least reliable bit partial sum takes its value in accordance with that.

In addition to that, five others special nodes were observed later in the code tree of a polar code [41]. The decoding trees of these nodes are shown in Fig. 2.10, where the node B is a R1 of length 2, SPC of length 4 or REP-SPC node of length 8 for the Type-I, the Type-II, and the Type-V node, respectively. Fast decoders for the SC algorithm were efficiently designed to decode these new special nodes. They are identified as follows:

— Type-I: In this node, all bits are frozen except for the last two ones which are information bits. The frozen bit sequence that corresponds to this node is: $\{0, \ldots, 1, 1\}$.

— Type-II: In this node, all bits are frozen except for the last three ones which are information bits. The frozen bit sequence that corresponds to this node is: $\{0, \ldots, 1, 1, 1\}$.

— Type-III: In this node, all bits are information except for the first two ones which are

frozen bits. The frozen bit sequence that corresponds to this node is: $\{0, 0, 1, \ldots, 1\}$.

— Type-IV: In this node, all bits are information except for the first three ones which are frozen bits. The frozen bit sequence that corresponds to this node is: $\{0, 0, 0, 1, \ldots, 1\}$.

— Type-V: In this node, all bits are frozen except for the last three and the fifth (from last) ones which are information bits. The frozen bit sequence that corresponds to this node is: $\{0, \ldots, 1, 0, 1, 1, 1\}$.

Furthermore, a generalization approach of merging special nodes was proposed in [22], in which the above-mentioned special nodes are merged to constitute multi-node subcodes and allow to apply fast decoding to larger subsets of bits. It follows that Type-I to Type-V nodes are particular cases of these generalized special nodes.

— Generalized REP node (G-REP) is a node whose descendants are all R0 nodes, except the rightmost one, that is a generic. This node is decoded using only the partial sums of the rightmost one and repeating it as many times as the number of the descendants R0 nodes.

— Generalized parity check node (G-PC) is a node whose descendants are all R1 nodes, except the leftmost one, that is R0. The decoding of this node is performed according to [42] by considering multiple parallel independent SPC nodes.

— Relaxed G-PC (RG-PC) node is a particular case of G-PC node in which additional frozen bits are present so that some of R1 nodes are in fact generic nodes whose rates are close to one. A sub-optimal decoding of this node that introduces a trade-off between error-correction performance and decoding latency is proposed in [22]. This node is decoded in the same manner as G-PC node since the additional frozen bits are ignored.

2.3.3 Fast SCL decoders

SSCL decoder

The tree representation in Fig. 2.9a corresponds to the polar code of length $N = 16$ and information block $K = 8$ denoted by PC(16,8). Each node includes $\boldsymbol{\alpha} = \{\alpha_0, \alpha_1, \ldots, \alpha_{N_\lambda - 1}\}$ LLR values and $\boldsymbol{\beta} = \{\beta_0, \beta_1, \ldots, \beta_{N_\lambda - 1}\}$ estimated bits i.e., partial-sums. At leaf nodes, the frozen and information bits are represented by white and black circles respectively. After computing path metrics, the decoding process searches the tree

for the L most reliable LLR subset. To do so, computed soft values α_l transit from parent to child nodes across the tree providing $\boldsymbol{\alpha}^{\mathrm{l}} = \{\alpha_0^{\mathrm{l}}, \alpha_1^{\mathrm{l}}, \ldots, \alpha_{\frac{N_\lambda}{2}-1}^{\mathrm{l}}\}$ to the left child node and $\boldsymbol{\alpha}^{\mathrm{r}} = \{\alpha_0^{\mathrm{r}}, \alpha_1^{\mathrm{r}}, \ldots, \alpha_{\frac{N_\lambda}{2}-1}^{\mathrm{r}}\}$ to the right child node by performing f and g operations respectively. Estimated bits i.e, partial sums β_l transit from child to parent and are computed by means of the received estimated bits of the left child node $\boldsymbol{\beta}^{\mathrm{l}} = \{\beta_0^{\mathrm{l}}, \beta_1^{\mathrm{l}}, \ldots, \beta_{\frac{N_\lambda}{2}-1}^{\mathrm{l}}\}$ and of the right child node $\boldsymbol{\beta}^{\mathrm{r}} = \{\beta_0^{\mathrm{r}}, \beta_1^{\mathrm{r}}, \ldots, \beta_{\frac{N_\lambda}{2}-1}^{\mathrm{r}}\}$ as follows:

$$\beta_i = \begin{cases} \beta_i^{\mathrm{l}} \oplus \beta_i^{\mathrm{r}}, & \text{if} \quad i < \frac{N_\lambda}{2}, \\ \beta_{i-\frac{N_\lambda}{2}}^{\mathrm{r}}, & \text{otherwise.} \end{cases} \tag{2.14}$$

The polar code of Fig. 2.9a features four constituent codes at stage $\lambda = 2$ which are R0, REP, R1 and SPC nodes all of length four. The pruned tree obtained following the identification of these constituent codes is provided in Fig. 2.9b.

A Simplified Successive Cancellation (SSCL) algorithm was proposed in [45] in order to reduce the time-complexity of the SCL algorithm by benefiting from the advantages of the previously-mentioned tree-pruning techniques. based on the idea of list sphere decoding [44], the SSCL algorithm proposes a computation of the PMs for three special nodes R0, REP, R1 from the LLRs of their top node. In addition to being much faster than SCL, the advantages of the SSCL algorithm include no perceived error-correction penalty and the reuse of the same sorter as the conventional SCL algorithm. With vectors α_l and β_l representing the LLR values at top of a R0, REP and R1 nodes, a hardware-friendly path metric computation for these nodes was proposed in [46]. The computation rules and the decoding of these special nodes in the case of list decoding successive cancellation follows:

$$\mathrm{PM}_{N_\lambda-1}^{l} = \frac{1}{2} \sum_{i=0}^{N_\lambda-1} \mathrm{sgn}\left(\alpha_{i_l}\right) \alpha_{i_l} - \alpha_{i_l}, \tag{2.15}$$

$$\mathrm{PM}_{N_\lambda-1_l} = \frac{1}{2} \sum_{i=0}^{N_\lambda-1} \mathrm{sgn}\left(\alpha_{i_l}\right) \alpha_{i_l} - \left(1 - 2\beta_{N_\lambda-1_l}\right) \alpha_{i_l}, \tag{2.16}$$

$$\mathrm{PM}_{N_\lambda-1_l} = \frac{1}{2} \sum_{i=0}^{N_\lambda-1} \mathrm{sgn}\left(\alpha_{i_l}\right) \alpha_{i_l} - \left(1 - 2\beta_{i_l}\right) \alpha_{i_l}, \tag{2.17}$$

where $1 - 2\beta_{N_\lambda-1_l}$ is the information bit estimate in the Rep node. A SSCL decoder achieves a decoding latency around 3.52 times smaller than the conventional SCL decoder

when decoding a PC(2048, 1024) optimized for $Eb/N_0 = 2$ dB [45].

SSCL-SPC decoder

An efficient way to decode SPC nodes was proposed in [42] as an extension to the SSCL decoder. It guarantees the preservation of the error-correction performance of the SCL decoder for a list size of 2 and entails a negligible degradation of $\leq$0.05 dB for other list sizes. Therefore, the pruned tree of the SSCL-SPC decoder becomes shorter by merging the Rep and R1 nodes of size 2 in the SSCL pruned tree into a single SPC node of size 4. A decoder architecture was designed in [42] with results showing an improved throughput by a factor of 3.16 at the cost of a 14.2% increase in area occupation when compared to the SSCL decoder. Following this technique, the computation of path metrics is obtained as follows: The least reliable bit which corresponds to the even-parity constraint is decoded first. In a SPC node of length N_λ, it is determined by

$$i_{\min} = \underset{0 \leq i < N_\lambda}{\arg\min}(|\alpha_i|), \tag{2.18}$$

and the parity is derived as

$$\gamma = \bigoplus_{i=0}^{N_\lambda - 1} \left(\frac{1}{2} \left(1 - \operatorname{sgn}\left(\alpha_i \right) \right) \right). \tag{2.19}$$

To satisfy the even-parity constraint, γ is computed for each path based on (2.19). The PMs are then initialized as

$$\mathrm{PM}_0 = \begin{cases} \mathrm{PM}_{-1} + |\alpha_{i_{\min}}|, & \text{if } \gamma = 1, \\ \mathrm{PM}_{-1}\,, & \text{otherwise.} \end{cases} \tag{2.20}$$

For the remaining parity check bits, the PM is updated by

$$\mathrm{PM}_i = \begin{cases} \mathrm{PM}_{i-1} + |\alpha_i| + (1 - 2\gamma)|\alpha_{i_{\min}}|, & \text{if } (1 - 2\beta_{i_l}) \neq \operatorname{sgn}\left(\alpha_i \right), \\ \mathrm{PM}_{i-1}\,, & \text{otherwise.} \end{cases} \tag{2.21}$$

Finally, when all the bits are estimated, the least reliable bit is set to preserve the even-parity constraint following

$$\beta_{i_{\min}} = \bigoplus_{\substack{i=0 \\ i \neq i_{\min}}}^{N_\lambda - 1} \beta_i. \tag{2.22}$$

Fast SSCL-SPC decoder

In the aim of reducing decoding latency and increasing the throughput, further special node decoding improvements were proposed in [43]. In fact, it was shown in [43] that the number of path splits required to decode R1 and SPC nodes can be further reduced and does not depend on the special nodes lengths anymore. The called Fast-SSCL and Fast-SSCL-SPC are an improved version of SSCL and SSCL-SPC decoders that result in faster decoders while keeping the error-correction performance unaltered. Therefore, in fast-SSCL decoding and list size L, the minimum number of path duplications in R1 nodes of length N_λ required to obtain the same results as the SSCL decoder is:

$$\min(L-1, N_\lambda). \tag{2.23}$$

Thus, any number of duplications smaller than this limit results in potential performance degradation. On the contrary, any number of duplications beyond this limit is redundant. The PM splitting is performed on the $\min(L-1, N_\lambda)$ least reliable bits of R1 nodes, i.e., those exhibiting the smallest absolute LLR values, the same way as they would in a SSCL decoder:

$$\mathrm{PM}_i^l = \begin{cases} \mathrm{PM}_{i-1}^l + |\alpha_{i_l}|, & \text{if } (1 - 2\beta_{i_l}) \neq \operatorname{sgn}(\alpha_i), \\ \mathrm{PM}_{i-1}^l, & \text{otherwise.} \end{cases} \tag{2.24}$$

The remaining bits are obtained through hard decision following:

$$\beta_i = \begin{cases} 0, & \text{if } \alpha_i \geq 0, \\ 1, & \text{otherwise.} \end{cases} \tag{2.25}$$

on the other hand, the number of path duplication in SPC node of length N_λ required to obtain the same results as the SSCL-SPC decoder is:

$$\min(L, N_\lambda). \tag{2.26}$$

If $\min(L-1, N_\lambda) = N_\lambda$ in the case of a R1 node and $\min(L, N_\lambda) = N_\lambda$ in the case of a SPC node. The total number of path forks is equal to the number of information bits in the node and the decoding process reverts to that of [45] and [42], respectively. However, in practical polar codes, the number of occurrences where $L-1 < N_\lambda$ and $L < N_\lambda$ for R1 and SPC nodes respectively, are often encountered. Therefore, using Fast-SSCL and Fast-SSCL-SPC algorithms, can significantly reduce the decoding latency in comparison with SSCL and SSCL-SPC algorithms, respectively.

A trade-off between error-correction performance and speed was also proposed in [43] where a new definition of the number of path splittings called S_{R1} and S_{SPC} generally smaller than the optimal number defined in (2.23) and (2.26) is applied in the decoding of R1 and SPC nodes, respectively. Simulation results of PC(1024 ,512) for L = 8 under Fast-SSCL, have shown a small degradation in error-correction (0.1 dB) at FER = $10^{-}5$ by choosing $S_{\text{R1}} = 1$. This error-correction performance gap with respect to the optimal value of $S_{\text{R1}} = 7$ was compensated for when choosing $S_{\text{R1}} = 2$. Similarly, the selection of $S_{\text{R1}} = 2$ and $S_{\text{SPC}} = 4$ under Fast-SSCL-SPC results in the same error-correction performance as the optimal values of $S_{\text{R1}} = 7$ and $S_{\text{SPC}} = 8$. Hardware architectures implementing both algorithms were proposed in the same work and have demonstrated a throughput of 1.86 Gb/s.

Fast SCL decoder for the new special nodes

Efficient decoders for the new identified special nodes, namely Type-I, Type-II, Type-III, Type-IV, and Type-V were proposed in [40] to improve the speed of SCL decoding and the afore-mentioned Fast SCL decoders without affecting the bit-error-rate and block-error-rate performance. Therefore, the Type-I to Type-V decoders require only 2, 2, $1 + \min(L-1, N_\lambda - 2)$, $1 + \min(L-1, N_\lambda - 4)$, and 2 clock cycles, respectively. These latencies are smaller than that of the Fast SCL decoders. For instance, a Type-III node is decoded in 8 clock cycles when implemented in hardware, while it is decoded in 29 clock cycles with Fast-SSCL-SPC decoder for $N_\lambda = 32$ and $L = 8$. Furthermore, the decoding of these nodes was extended to SC flip (SCF) decoders in [7].

However, since all these nodes include at least two information bits, their decoding requires the estimation of multiple bits at a time. Specifically, Type-I needs to estimate two bits at the same time, Type-II needs to estimate three bits and Type-V needs to estimate four bits. At each decoding step, they produce $4L$, $8L$, and $16L$ codeword candidates, respectively, before selecting L. In this case, a larger sorter is required when implementing

Table 2.1 – Information, encoded block and mother polar code lengths supported by polar coding in the NR physical channels.

Physical channel	Supported information block lengths	Supported encoded block lengths	Supported polar code lengths
Physical Uplink Control Channel (PUCCH)	$A \in [12, 1706]$	See equation (2.27)	32, 64, 128, 256, 512, 1024
Physical Broadcast Channel	32	864	512
Physical Uplink Control Channel (PUCCH)	$A \in [12, 140]$	$G \in [A + 24, 8192]$	32, 64, 128, 256, 512

the decoder in hardware. In addition, two and four parallel SPC decoders are required to decode Type-III and Type-IV nodes, respectively, [40].

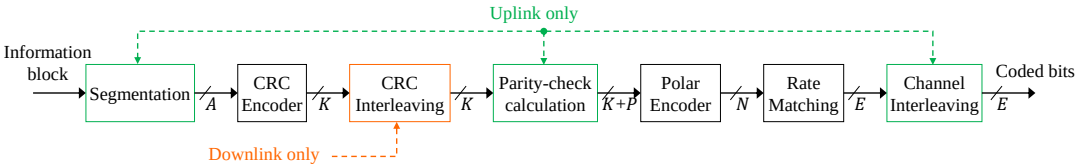
2.4 The polar code of 3GPP 5G NR

In this section we provide a short overview of the encoding process for the 5G NR polar code following the block diagrams of Fig. 2.11a and Fig. 2.11b. This code family was adopted as forward error correction for the Uplink and the Downlink Control Information (UCI and DCI) . More precisely, it is used over the Physical Uplink Control/Shared Channel (PUCCH/PUSCH), the physical downlink control channel and the Physical Broadcast Channel (PDCCH/PBCH). In Fig. 2.11a and Fig. 2.11b, the component in orange applies only for downlink while those in green apply only to uplink. The supported information block lengths, encoded block lengths and mother polar code lengths by the 5G standard are summarized in Table 2.1. Bounds on the encoded block length for the UCI are given by:

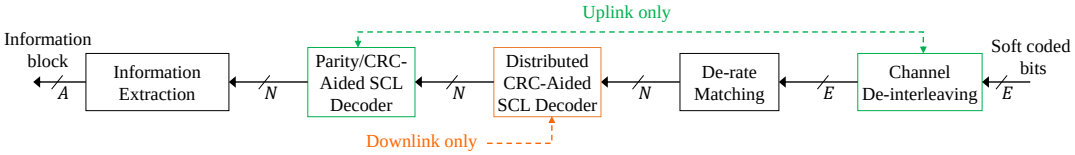
$$E \in \begin{cases} [A + 9, 8192], & \text{if } A \in [12, 19], \\ [A + 11, 8192], & \text{if } A \in [20, 359], \\ [A + 11, 16385], & \text{if A} \in [360, 1012], \\ [2[A/2] + 22, 16385], & \text{if A} \in [1013, 1706]. \end{cases} \tag{2.27}$$

2.4.1 CRC-bits attachment, scrambling and interleaving

L_{CRC} CRC bits are appended to the information sequence to allow error detection and improve the performance of the list polar decoder. Three different CRC generator



(a) Block diagram for channel encoding in 3GPP 5G NR PUCCH/PUSCH.



(b) Block diagram for channel decoding in 3GPP 5G NR PDCCH.

Figure 2.11 – The 3GPP 5G NR Polar coding and decoding chain.

polynomials were carefully chosen for the various physical control channels as follows:

$$\begin{aligned} g_6(x) &= x^6 + x^5 + 1. \\ g_{11}(x) &= x^{11} + x^{10} + x^9 + x^5 + 1. \\ g_{24}(x) &= x^{24} + x^{23} + x^{21} + x^{20} + x^{17} + x^{15} + x^{13} + x^{12} + x^8 + x^4 + x^2 + x + 1. \end{aligned} \tag{2.28}$$

The polynomials g_6 and g_{11} are used in PUCCH when the information block length is $A \in [12, 19]$ and $A \in [20, 1706]$ bits, respectively. In the case of PDCCH and PBCH a larger number of CRC bits is needed to enable early termination in the case of failures and to reduce the incidence of false alarms during blind decoding [23]. Therefore, g_{24} with $L_{\text{CRC}} = 24$ CRC bits are used. Following their computation, the last 16 CRC bits are scrambled by performing XORs with the Radio Network Temporary Identifier (RNTI) as defined in [3]. Moreover only in the downlink case, CRC interleaving is employed following CRC attachment and scrambling. The CRC bits are then shuffled according to a specific interleaving pattern of length $A_{\max} + L_{\text{CRC}} = 164$ bits. All interleaver patterns for $A \leq A_{\max}$ may be derived from this single mother pattern. The CRC interleaver is designed to ensure that each CRC bit depends only on the previously decoded information bits. This allows the CRC bit value to be calculated without waiting for the following information bits in the interleaved sequence. Hence, the CRC check operation can be performed early during decoding allowing an early termination when it fails. However, when at least one candidate path verifies the CRC check, the inverse interleaving pattern is applied to restore the original order of the vector including information and included CRC bits. A final CRC check may be performed at the end of the decoding process. To

do so, the $A + L_{\text{CRC}}$ decoded bits are sent to the CRC decoder and the last L_{CRC}-bits of the obtained syndrome are compared to L_{CRC} zero-valued bits to verify the check pass.

2.4.2 Sub-channel allocation and bits insertion

For the NR polar code, a so called universal reliability sequence $Q_0^{N_{\max}-1}$ consisting of a list of integers between 0 and $N_{\max} - 1$, with $N_{\max} = 1024$, sorted in reliability order is provided. Included integer values correspond to the reliabilities of the sub-channels together with the rate-matching scheme. It is used in order to determine the set of the frozen, information, CRC and PC bit positions. The set of frozen bits is first identified. It consists basically of the non transmitted bits removed by the rate-matching scheme and the remaining least reliable bits. Next, the information and CRC bits are placed in the most reliable sub-channel positions. The PC bits that are used in the PUCCH when $A \in [12, 19]$ take the remaining most reliable positions. The sub-channel allocation process is detailed in [17]. The computation of the PC bits is obtained through a length-5 cyclic shift register [90]. The calculation of the PC bits is obtained through a PC bit generator consisting of a length-5 cyclic shift register. On the decoder side, the determination of frozen, info, CRC and PC bits positions follow the same process that is used with the encoding. Unlike frozen bits which are already known by the decoder, the PC bits take their values from the PC bit generator.

2.4.3 Rate matching

The obtained vector u following CRC bits attachment and insertion of information bits is encoded to vector x of length N following $x = u \cdot G_N^{\otimes n}$, where $G_N^{\otimes n}$ is the n^{th} Kronecker power of $\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$.

The first step of rate matching consists in dividing the encoded vector x into 32 sub-blocks, each of length $N/32$ then applying the sub-block interleaver. The interleaved vector is then subject to three different types of rate matching to obtain the encoded block length G from the mother polar code length N through a circular buffer. If $G \leq N$, the encoded bit vector x is either punctured when $R \leq \frac{7}{16}$ and the first $N - G$ bits are not transmitted, or shortened when $R > \frac{7}{16}$ and the last $N - E$ bits are not transmitted. Otherwise the vector y is repeated by transmitting twice the first $N - G$ bits. The received LLRs are obtained depending on the applied rate-matching scheme before transmission.

Hence, the LLRs corresponding to the repeated bits are accumulated, those corresponding to punctured bits are set to zero and finally, those used for shortening are set to infinity and appended to the set of G received LLRs. A final step of channel interleaving concerning PUCCH and PUSCH UCIs is performed to improve the error correction capability of the polar code when associated with high order modulation schemes.

2.5 Summary

In this chapter we provided an overview of polar codes including the polarization concept on which they are based and the way they are constructed. A special focus was made on the decoding algorithms and in particular on the low-complexity successive cancellation decoding. Based on different binary-tree representations, tree-pruning techniques at the origin of various simplified polar decoders were presented, motivated by their positive impact on latency and throughput. Finally, in the aim of developing an efficient polar decoder for the recently adopted 5G NR polar codes, an overview of their specific encoding and decoding processes was provided at the end of this chapter.

Chapter 3

Design space exploration for polar decoders

Contents

The list-augmented SC decoding algorithm is best suited to decode the polar codes of the 5G NR control channel mainly thanks to its good error-correction performance when aided with outer codes such as CRC and parity bits. However, finding the best suitable hardware architecture for decoding is not trivial, particularly under the stringent requirements to comply with block length and code rate flexibilities while maintaining low decoding latency and hardware complexity.

This chapter presents our first major contribution related to design space exploration and concerning the study of the impact of main code and decoder design parameters on latency, throughput and the hardware complexity of semi-parallel decoding architectures. The impact of these parameters on the hardware efficiency of semi-parallel architectures is significant. Therefore, we propose two multi-frame decoding approaches that increase the throughput and improve the utilization rate of the processing units of these architectures. Detailed analytical and logic synthesis results are provided and compared for a large range of relevant code parameters in order to draw general tendencies towards building a framework for implementing flexible yet efficient FEC decoders for polar codes. Furthermore, a complete software simulation environment of polar coding/decoding is proposed for performance evaluation under different algorithms in both floating-point and fixed-point data representation. The chapter is organized as follows. Section 3.1 presents the developed software simulator, its features, and the performance of the 5G NR polar codes using various decoding algorithms and several quantization levels of the decoder metrics. Section 3.2 presents the hardware architectures used to implement polar codes, together with the algorithmic and architectural parameters considered in the proposed study. Section 3.3 provides latency performance analysis. Hardware complexity results and throughput analysis are discussed in Section 3.4, while the hardware efficiency analysis of semi-parallel decoding architectures together with two multi-frame decoding approaches are provided in Section 3.5. Finally, Section 3.6 concludes the chapter.

3.1 Performance of 5G NR polar codes

In this section, we propose to study the error-correcting performance of 5G NR polar codes in terms of Bit-Error-Rate (BER) and Frame-Error-Rate (FER). The simplified

family of algorithms based on tree-pruning techniques has shown significant benefits in reducing the latency of the SCL decoders, which makes it particularly suitable for low latency and low complexity implementation of the control channel polar decoders. The performance of the underlying algorithms with list-decoding is evaluated and compared to the performance of the SCL decoder for a list size of eight. To prevent the error-correction performance of polar codes from degradation at hardware level design, several quantization levels for message passing and internal metrics, namely LLRs, and PMs, are studied. For this, both fixed-point and floating-point simulation performance of the decoder are compared. A software simulator that integrates the entire encoding and decoding schemes of polar codes as described in [3] is designed and simulation results are reported accordingly.

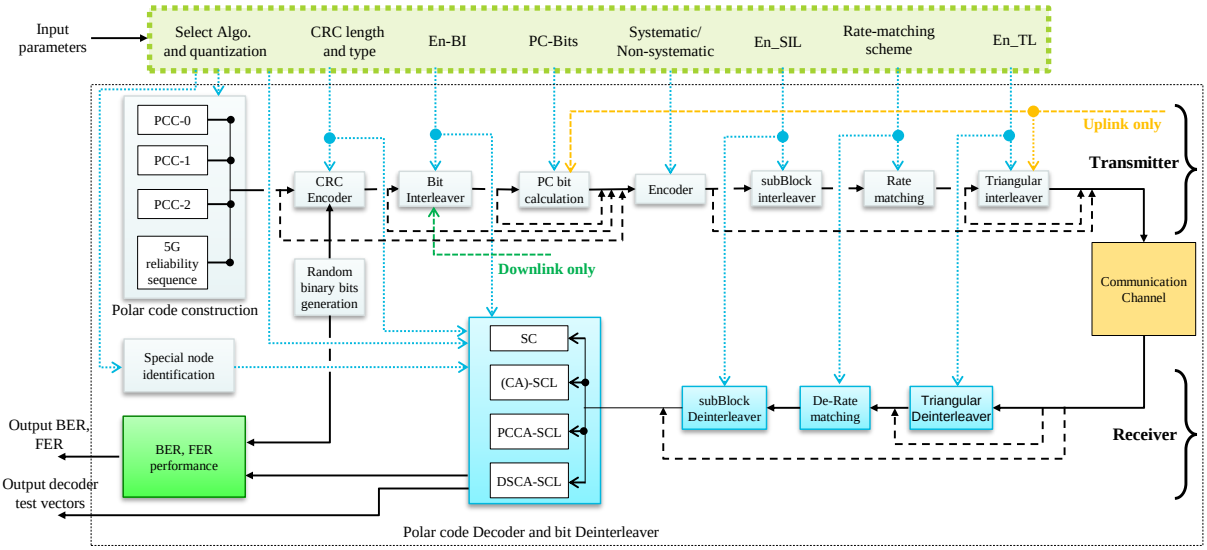


Figure 3.1 – Block diagram of the developed software simulator for polar codes.

3.1.1 Proposed polar code simulator

A software simulator for polar codes is designed to evaluate performance corresponding to efficient hardware decoders for polar codes. The block diagram of this simulator is represented in Fig. 3.1. It integrates all the components of a digital communication system model [81], excluding the source encoding/decoding, and provides several simulation choices. The software simulator is composed of three major parts: transmitter, communication channel, and receiver. The transmitter generates random messages, encodes them according to specific schemes, maps them onto a constellation, then sends them over the desired

communication channel model. The applied channel function depends on the channel type. Typically an Additive White Gaussian Noise (AWGN) is applied. In the receiver part, received messages are de-mapped from the chosen constellation and decoded according to a particular algorithm selected in advance. Finally, the BER and FER performance of the decoder is performed based on the original and decoded codewords.

In addition to that, the developed simulator includes a plurality of features and choices at the algorithmic level, in code construction methods, and in using specific interleaving and rate matching operations. These features are detailed in Table 3.1. Therefore, the classical and most commonly polar code construction methods [53, 59, 85, 92] presented in [89], namely PCC-0, PCC-1, and PCC-2, are included to divide bit-channels into those that are good and those that are bad. However, in the context of 5G NR polar codes, a reliability sequence provided in the 3GPP standard is used in the sub-channel allocation process instead of using polar code construction algorithms. Furthermore, using algorithms that rely on tree-pruning techniques, a list of special nodes is created based on the frozen set generated in a previous step. Various tree-pruning options are considered depending on the choice of the algorithm and the maximum size of the special nodes defined in advance.

Alongside the simple encoding operation of polar codes, the 5G NR polar code complete encoding and decoding chain is integrated into this simulator. Moreover, outer codes such as CRC and parity check codes are included and can be used in association with the polar code depending on the selected encoding scheme. The calculation of CRC bits is performed using cyclic generator polynomials. In addition to those defined in (2.28), other polynomials and other options for initializing the CRC bits are available, allowing the error-detection capability to be evaluated under different polynomials. Parity bits are calculated in three different forms, as described in [3].

The information and the encoded bits can be subject to different interleaving and rate matching types, particularly in the 5G NR control channel. Therefore, Three interleaving operations, called bit interleaving, sub-block interleaving, and channel interleaving, are included in the encoding process. Also, three types of rate matching procedures are included: puncturing, shortening, and repetition. At the receiver, the reverse operations of interleaving and rate matching are performed first if already processed during the encoding scheme. These operations can be applied independently on both transmitter and receiver sides. This provides flexibility in running various simulation types for assessing the performance of polar codes not limited to the ones in 5G NR. However, in the case of the 5G NR polar codes of the control channel, all these operations are systematically applied to

Table 3.1 – Polar code simulator features.

Initialization	**Encoding operations**	**Decoding algorithms**
• PCC-0 • PCC-1 • PCC-2 • **5G reliability sequence** • **List of special nodes**	• **CRC bit computation** • Bit Interleaving • PC generation • **Systematic encoding** • Non-Systematic encoding • **Sub-block Interleaving** • **Rate Matching** • Channel Interleaving • Triangular De-Interleaver • **De-Rate-Matching** • **Sub-block De-Interleaving**	• SC • SCL • CA-SCL • Bit De-Interleaver • PC-SCL • **SSCL decoding** • SSCL-SPC decoding • Fast-SSCL decoding • Fast-SSCL-SPC decoding • Various code length N • Various list size L • **quantization schemes** • **Hardware**/algorithmic design approach • **FER and BER computation**

match the specified encoding and decoding schemes. A unique variable is used to select the type of physical channel used for the simulation. Moreover, to increase the flexibility of the proposed simulator, all the integrated functions apply an easily modifiable programming method based on C language structures. Therefore, it provides better portability and handling of the simulator and facilitates the integration of additional functions. On top of that, it facilitates the validation of any encoding and decoding processes, in addition to those from the 5G NR polar code.

The received soft message is then decoded using one of the various implemented polar code decoders, which are summarized in Table 3.1. Furthermore, thanks to a hardware-oriented design of the decoding algorithms, the software polar code simulator constitutes a powerful tool to produce and support the validation of efficient hardware architectures for polar codes. Indeed, memory pointers are preferred to lazy copying techniques of the decoder data when candidate competition is involved in the list-decoding of polar codes. This design approach leads to producing software polar decoders that come as close as possible to the hardware version of these decoders, which simplifies their validation process by providing reference test vectors. A study was carried out using the proposed simulator

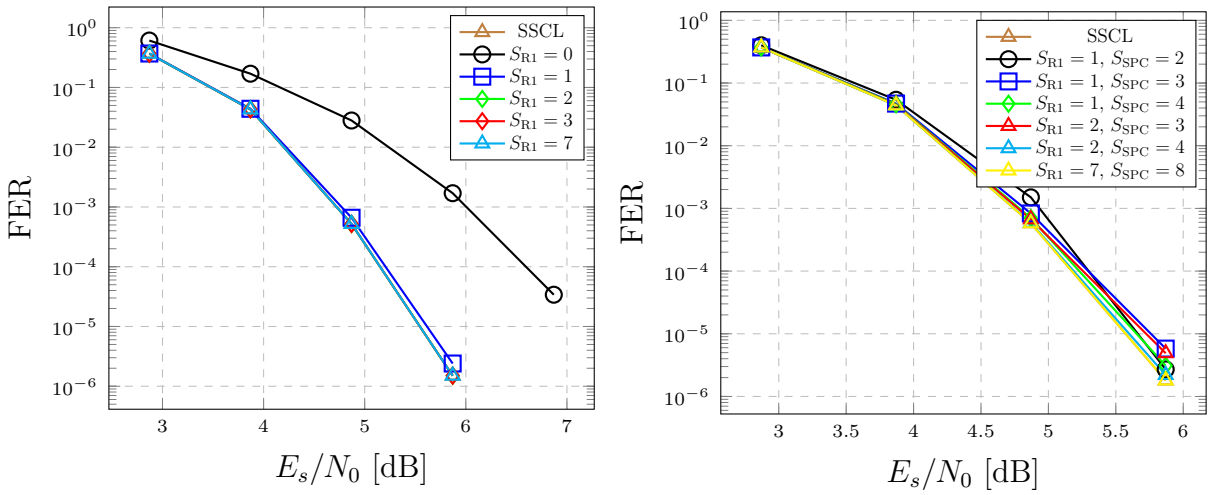
to evaluate the impact of data quantization on the error-correcting performance of polar codes. It offers a wide choice of quantization schemes thanks to a fixed-point representation of the decoder data (channel LLRs, internal LLRs, and path metrics). It performs FER and BER calculations on both floating-point and fixed-point decoders. Finally, thanks to the portability of this simulator, all developed features are programmed independently and can be easily extracted to be used for any future development purpose. An example of using a group of functions and operations (in **bold characters**) to perform a FER simulation of a polar code is shown in Table 3.1.

3.1.2 Performance of 5G NR polar codes with tree-pruning decoders

A trade-off between the error-correcting performance and the speed of decoders is proposed in [43] based on reducing the number of path splittings S_{R1} and S_{SPC} when decoding R1 and SPC nodes. It was shown that only a small performance degradation is reported with few combinations of small values of S_{R1} and S_{SPC}. To expand this observation to the polar codes that have been specified for the 5G NR control channel, we propose to evaluate the impact of varying S_{R1} and S_{SPC} below their optimal values on the performance of the uplink PC(256,128). Fig. 3.2a shows the FER performance comparison of the Fast-SSCL decoder for $L = 8$ and different values of $S_{\text{R1}} = 0, 1, 3, L-1$. Therefore, reducing the number of maximum path splittings in R1 nodes from $S_{\text{R1}} = 7$ to $S_{\text{R1}} = 2$ does not seem to introduce any loss in error correction. However, we notice a small loss in performance less than 0.1 dB at 10^{-5} with $S_{\text{R1}} = L-1$ compared to the optimal Fast-SCCL decoder, i.e., $S_{\text{R1}} = L-1$. In Fig. 3.2b, we show the FER performance comparison of the Fast-SSCL-SPC decoder for $L = 8$ and different values of $S_{\text{R1}} = 1, 2, L-1$ and $S_{\text{SPC}} = 2, 3, 4, L$. For S_{R1} values larger than two and S_{SPC} values larger than 4, the Fast-SSCL-SPC decoder provides the same FER compared to the one using the optimal values of $S_{\text{R1}} = L-1$ and $S_{\text{SPC}} = L$. However, a small degradation in the FER performance is seen for values lower than these.

3.1.3 Impact of quantization on the performance

Different quantization schemes are used during the implementation of hardware architectures for SC and SCL decoders. In [16], to avoid saturation of the LLR values, the number of bit quantization is increased by one bit at each decoding stage. In this way,



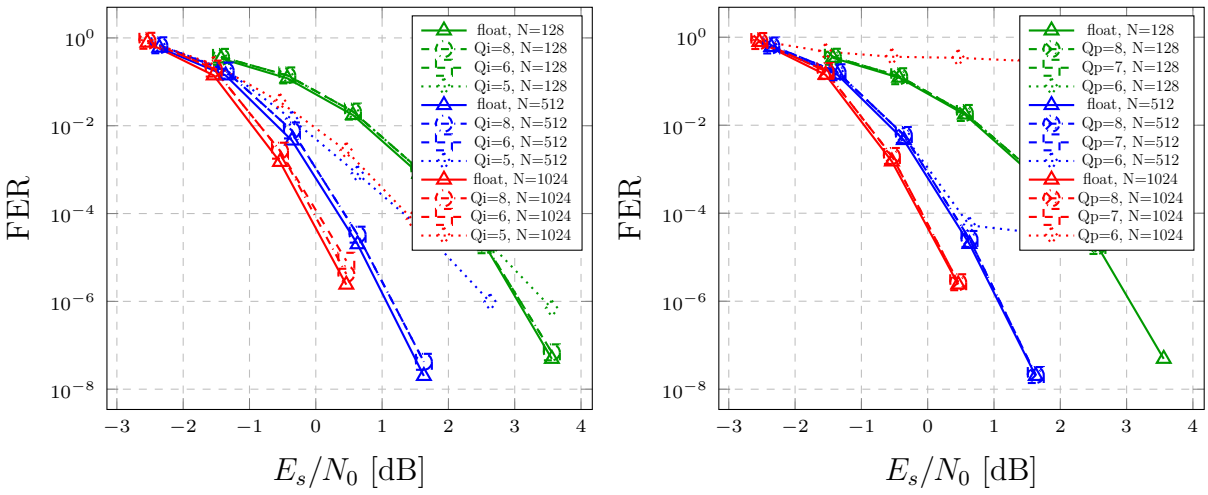
(a) Fast-SSCL

(b) Fast-SSCL-SPC

Figure 3.2 – FER performance comparison of Fast-SSCL and Fast-SSCL-SPC decoders of PC(256,128) for $L = 8$ and different values of S_{R1} and S_{SPC}.

the quantization of the channel LLRs is the only source of performance degradation in comparison to the floating-point implementation. The robustness of this technique comes at the cost of a large number of bits and additional complexity in the processing elements. However, All the LLRs in [13] use a 6-bits uniform quantization with a constant step size $\Delta = 1$ between the different level of quantization step size, while 8-bits are set to represent the PM values. A Memory efficient quantization technique is proposed in [62] to reduce the number of bits to store in the initial decoding stages at the cost of a slight error performance degradation. In [77], the authors proposed a logarithmic non-uniform quantization scheme using look-up tables which outperforms the uniform quantization at any code length and provides an error-correcting performance close to the floating-point case.

In the aim of producing an efficient hardware architecture to decode the 5G NR polar code, we propose to study the effect of quantization on the performance of these codes. Several quantization levels of the received LLRs, internal LLRs, and PMs metrics are investigated, and the FER performance of the quantized decoder is compared to the floating-point one. Therefore, We assume a sign and magnitude (SM) representation of LLR values with one bit dedicated to the sign of the LLR. However, since PMs are positive real values, all the quantization bits are dedicated to represent the magnitude. Furthermore, we propose to represent LLRs and PMs following a uniform quantization

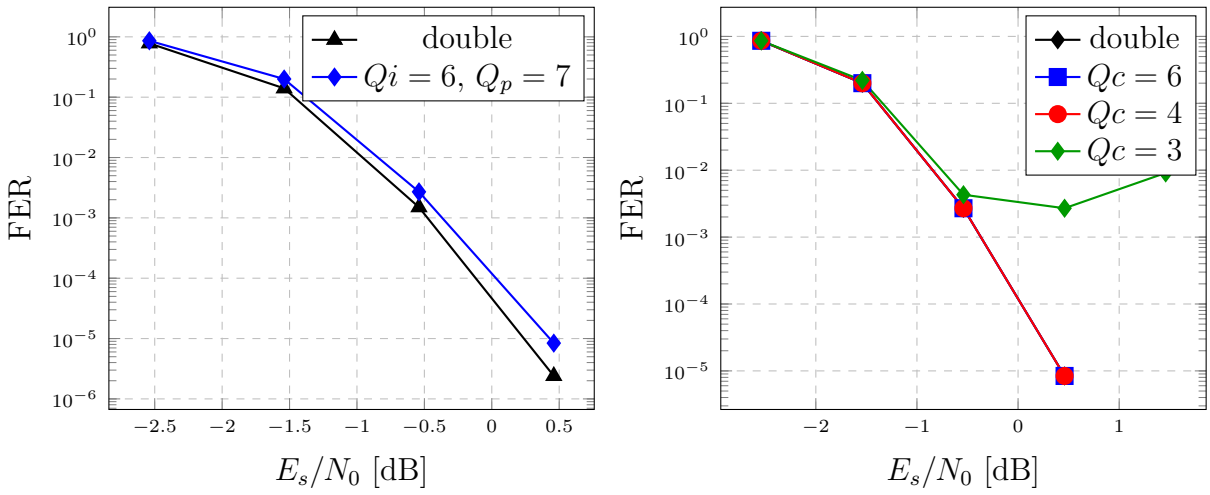


(a) Effect of LLR quantization.

(b) Effect of PM quantization

Figure 3.3 – Effect of LLR and PM quantizations on the error-correction performance of three different PUCCH polar codes of lengths $N = 1024, 512$ and 128.

scheme with a step size $\Delta = 1$. We denote by (Q_i, Q_c, Q_p) the quantization format where Q_c is the number of bits to represent channel LLR values. Q_i is the number of bits to represent the internal LLR values and Q_p is the number of bits to represent path metrics values. In Fig. 3.3a and Fig. 3.3b, we show the effect of LLR quantization and PM quantization on error-correction of three different PUCCH polar codes of lengths $N = 1024, 512$, and 128 bits. Note that when LLRs are quantized, PMs are represented in floating-point and vice-versa. Three values of Q_i are used in this study, $Q_i = 5, 6$ and 8, and channel LLRs are quantized with the same number of bits as internal LLRs, $Q_i = Q_c$. And three values of Q_p, $Q_p = 6, 7$ and 8. We can clearly identify a relationship between quantization and polar code length N. Fig. 3.3a and Fig. 3.3b show that the number of LLRs and PMs quantization bits that guarantee a negligible degradation in the FER performance of polar codes depends directly on the code length. Indeed, a number of quantization bits that yield a good error-correction of a given length are insufficient to yield the same error-correction of a larger length. This is because longer codes require more decoding stages. Note that LLR values keep increasing from one stage to another. At 10^{-5} of FER, only 5-bit quantization is sufficient to preserve the error-correction capability of polar codes of length 128 compared to the floating-point decoder. However, one more bit, i.e., 6 bits, is required with polar codes of length $N = 512$ to keep the FER performance within 0.1 dB of the floating-point decoder. The same loss in performance occurs with



(a) combined $Q_i = 6$ and $Q_p = 7$ quantization.

(b) Effect of channel LLR quantization

Figure 3.4 – Effect of the selected LLR and PM quantization levels and of the channel quantization on the error-correction performance of a PUCCH 1024 polar code.

polar codes of length $N = 1024$ when using 7 or 8 bits of quantization. This small loss of performance will slightly increase to 0.19 dB for $N = 1024$ while using only 6 bits of quantization. For code lengths lower than $N = 256$, no loss in performance is noticed. However, for each code length, the performance starts to diverge from the floating-point simulations at a specific E_s/N_0 where the range of an internal LLR increases. For instance, the performance of the polar code of length $N = 128$ starts to diverge at $E_s/N_0 = 2.6$ dB with $Q_i = 5$. Increasing Q_i to 6 will guarantee that all polar codes of lengths $N \leq 1024$ perform closely to the floating-point case at FER $= 10^{-5}$ and lower. On the other hand, Fig. 3.3b shows that when LLRs are represented in floating-point, 7 bits are sufficient to represent the PM values. A parallel analysis of the received channel LLRs have shown that their range of values is small compared to internal LLR values, especially at low decoding stages. Indeed, their dynamic range increase when moving towards stage 0 as multiple additions of LLR pairs are performed by g function. Consequently, there is a potential interest in further reducing the number of quantization bits of the channel LLRs. In this case, two distinct memories are required to store the internal and the channel LLRs at the hardware design level. Fig. 3.4b shows the effect of channel LLR quantization on the error-correcting performance of a PUCCH polar code of length $N = 1024, 512$ with $Q_i = 6$ and $Q_p = 7$. Therefore, reducing the number of channel LLR quantization bits from $Q_c = Q_i = 6$ to $Q_c = 4$ does not introduce any performance loss. Consequently,

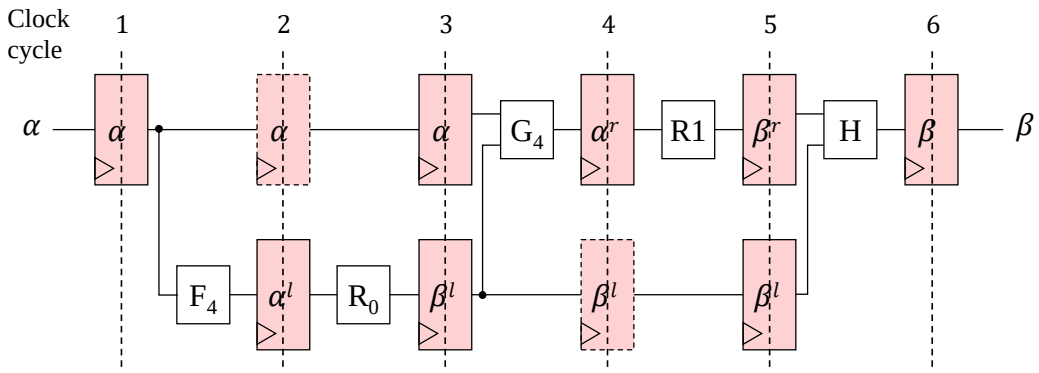


Figure 3.5 – Fully-unrolled deeply-pipelined decoder for a (8, 5) polar code.

the (6,4,7) quantization scheme is the minimum number of integer quantization bits that yields performance around 0.1 dB from the floating-point decoder.

3.2 Hardware architectures

In the last decade, multiple hardware implementations dedicated to decode polar codes have been proposed for both FPGA and ASIC targets [14, 28, 43, 66, 68, 93, 99] Two main architecture models are investigated in the literature: unrolled and semi-parallel architectures.

3.2.1 Unrolled architectures

Unrolled architecture model consists in allocating a dedicated hardware resource for each of the operations that occur during the decoding process. This allows the decoder to process multiple frames simultaneously. Indeed, A deeply-pipelined fully-unrolled architecture is able to output one decoded frame at every clock cycle by introducing pipeline stages between the operations. Consequently, such decoder architecture can reach hundreds of Gbps of throughput on ASIC technology at the cost of high memory requirements. However, the quadratic increase with code length of the hardware resources used to support this kind of parallelism leads to a high-complexity decoders [37]. An example of a deeply-pipelined decoder of a PC(8,3) is illustrated in Fig. 3.5. The internal LLRs and partial sums vectors denoted by α and β are stored in the registers illustrated in light red. The white blocks are the operations performed by the decoder. The output of each operation is stored in pipeline registers as long as they are needed by further functions.

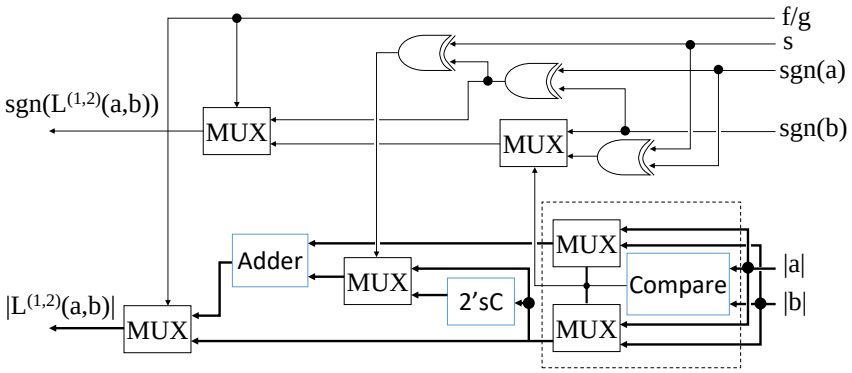


Figure 3.6 – Processing element architecture.

For instance, two registers β^l are needed to retain the estimate bits during 4th and 5th clock cycles and one register α_l is needed to store the left-child LLR that are fed to the R0 node. At every clock cycle a new frame is loaded in the most left register and a codeword is output, i.e., the architecture is decoding as many codeword as there are pipeline stages. This kind of unrolled architectures is described in [37]. A compromise between high throughput and memory reduction is proposed in [38] where a partially-pipelined architecture is obtained by removing the dashed register which reduce the required memory. The main idea behind unrolling a decoder is to increase its throughput. However, this results in limited length and rate flexibility implementations, in particular when tree-pruning algorithms are used to decode polar codes. Indeed, bringing a small change to the location of the information and the parity bits withing the frozen set of the polar code leads to a completely different list of special node types and sizes. This limits their suitability for low-latency flexible 5G NR polar decoders.

3.2.2 Semi-parallel architectures

Semi-parallel architecture model consists in integrating a certain number of processing elements (PEs) dedicated to compute a single or multiple operation types regardless of the length or the rate of the targeted set of polar codes. In the case where the number of operations to perform in parallel at a given time is greater than the instantiated PEs, these operations are scheduled in sub-groups to be processed sequentially. Therefore, the achievable throughput is typically lower than that of the unrolled fully-parallel architecture model. However, semi-parallel architectures enable the use of specific hardware optimizations such as arithmetic resource sharing and memory access sharing. Considering the

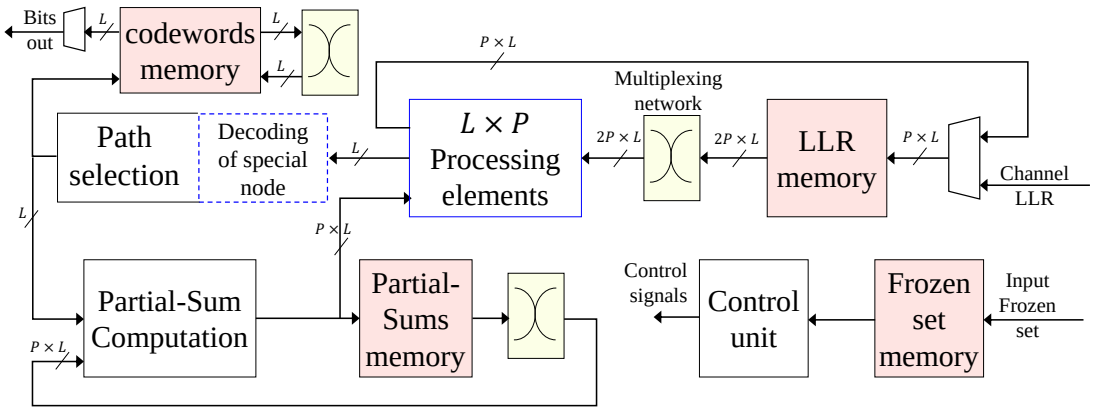


Figure 3.7 – Semi-parallel architecture model for SCL decoders.

flexibility requirement, this leads to improved hardware efficiency. Due to the sequential decoding nature of SC algorithm, its two main operations f and g cannot be overlapped and are always performed in two distinct time periods. Thus, an area-efficient combined processing element is proposed in [55] in which these two operations are carried out by a single PE that exploits resource sharing and can perform both operations alternately. The architecture of a processing element is depicted in Fig. 3.6. The LLR values are in sign-magnitude (SM) format. a compare and select (CS) component is used as a first step to either support the minimum search operation needed for performing f or to support the signed addition/subtraction performed by g, three xor logical gates are used to provide the sign of the output LLR and determine whether an addition or subtraction has to be carried out by the adder during the g operation.

Semi-parallel decoder architectures are scalable and offer a broad spectrum of algorithmic and architectural options to explore and to adapt depending on the desired performance and hardware limits imposed at the implementation level. This makes them a first choice for a flexible implementation. The semi-parallel architecture model for SCL decoders used in this study is depicted in Fig. 3.7. It includes a set of processing elements, path selection unit and partial sum computation unit. The architecture comprises in addition multiplexing networks to interface between memories and computation units. A control unit is used to produce all control signals required in the decoding process. This architecture model may include in addition a unit to decode special nodes when they are considered (dotted block). Four memory blocks store the LLRs, partial sums, decoded codewords and the frozen set are used in the proposed semi-parallel architecture (red colored blocks). The purpose of these memories is:

— LLR memory: the decoding relies on dedicated memories to keep the LLR values available for the computation units as long as they are needed. Therefore, up to $N-1$ LLR values are needed to be kept in memory during the decoding process. In some SC and SCL decoder implementations, input Channel and internal LLRs are stored in two distinct memories. First to avoid duplicating the memory for storing channel LLRs since they are common to all the L path of an SCL decoder. Second, to reduce the memory footprint by reducing their quantization bits.

— Partial sum memory: The PE must be provided with partial sums as part of g function computation. It was found that $N-1$ bit registers are sufficient for that. However, RAM-based memory are also used to read and store PS without lowering the throughput.

— Codewords memory: A $N \times L$ memory is used to store the codewords bits as and when decoded.

— Frozen bit memory : A $N-$bits length or an equivalent list of special nodes is required to store the frozen set or the special nodes of a specific polar codes when tree-pruning techniques are used.

For SCL semi-parallel decoder, the lazy copy technique that occur after path selection can be replaced with pointers and require a further memory to keep track of the survived paths after candidate competition.

3.2.3 Architectural and algorithmic parameters

Targeting a semi-parallel architecture model to design low-latency flexible polar decoders, the impact of several algorithmic and architectural parameters needs to be investigated. The main parameters that are considered in this work are listed below:

— Number of instantiated PEs: The number of processing elements is a main architectural parameter of the semi-parallel architecture model. Selecting the right number is not straightforward and needs a thorough analysis with respect to the other flexibility parameters.

— Tree-pruning techniques: Identifying special nodes in the polar decoder tree and applying tree-pruning techniques with corresponding special decoding algorithms impact significantly the performance metrics of the polar decoder. The influence of different pruning techniques cited in Section 2.3.2 are analysed in this work.

— Code length N: A length-flexible implementation increases the complexity of the decoder. Furthermore, the code length may significantly alter the influence of the the above-mentioned parameters on the performance metrics of the polar decoder. For this parameter, the following lengths specified in 5G NR will be considered: $N = 64$, 128, 256, 512 and 1024.

— Code rate R: A rate-flexible implementation decreases the hardware efficiency of the decoding architecture especially when supporting a wide range of values. In this work, we consider the polar codes of PUCCH 5G NR with R ranging from 1/8 to 5/6.

Typical values, convenient for 5G NR polar codes, are considered for other parameters such as data format, quantization scheme, and list size for SCL algorithms. LLR values are represented in sign and magnitude (SM) format and quantized on five and seven bits in PEs and special nodes decoders, respectively, whereas the list lize L is set to eight.

3.3 Latency analysis

In this section, we propose to evaluate the decoding latency of the 5G NR polar codes under the architectural and algorithmic parameters defined in the previous section. For the upcoming equations of latency and for the analytical results we assume that each elementary operation of the decoding process requires one time-step. Therefore, the reported number of clock cycles (CC) required to decode a single frame corresponds to that number of time-steps, which represents the latency. Following this assumption, the latency of decoding one codeword of length N with K information bits using SC on semi-parallel (SP) architecture can be expressed as:

$$\begin{aligned}\mathbb{L}_{\mathrm{SC}}^{\mathrm{SP}} &= \underbrace{\sum_{j=0}^{p} 2^{n-j}}_{\text{NON-AFFECTED STAGES}} + \underbrace{\sum_{j=p+1}^{n-1} 2^{n-j}2^{j-p}}_{\text{AFFECTED STAGES}}. \\ &= 2N + \frac{N}{P}\log\left(\frac{N}{4P}\right)\end{aligned} \tag{3.1}$$

where $p = \log_2 P$ and $n = \log_2 N$. This expression is derived from (2.12) by taking into consideration both affected and non-affected decoding stages j by the introduction of P PEs [55]. For an SCL decoder that comprises P PEs per list, this latency is increased by K as path selection needs to be performed K times [14].

However, if fast decoding techniques relying on decoding simple constituent codes are used, the latency formula will be sensitive to any change of the rate R. Therefore, assuming having the size of all the constituent codes defined for a given N and R, we define E such as $E = \{E_0, E_1, \ldots, E_{\log_2 M-1}\}$ is the set whose element E_m, $0 \leq m \leq \log_2 M - 1$, represents the number of constituent codes of length 2^{m+1} and M is the size of the largest constituent codes that are considered during the tree-pruning technique. Therefore, to derive the latency of the Pruned Decoder (PD) on a semi-parallel architecture, we should remove the latency of traversing the sub-trees corresponding to the identified special nodes (constituent codes) from the latency of the semi-parallel decoder provided in (3.1) for $N = 2^n$. This latency reduction can be computed through (2.10) with $N = 2^{m+1}$ for special nodes that satisfy $m \leq p$, and through (3.1) with $N = 2^{m+1}$ for the remaining special nodes. We should also add the latency that is required to decode each of the identified special nodes and add K' which is the number of remaining information bits that do not constitute special nodes. Thus, the latency of the SCL PD semi-parallel decoder can be expressed as:

$$
\begin{aligned}
\mathbb{L}^{\mathrm{SP}}_{\mathrm{SCL\ PD}} &= \mathbb{L}^{\mathrm{SP}}_{\mathrm{SC}} + \mathbb{L}^{\mathrm{SN}}_{\mathrm{SCL}} + K' - \Bigg(\mathbf{2} \cdot E_{\mathbf{0}} + (\mathbf{4{+}2}) \cdot E_{\mathbf{1}} + \ldots + E_{\mathbf{p}} \cdot \sum_{k=0}^{\mathbf{p}} 2^{\mathbf{p}-k+1} \\
&\quad + E_{\mathbf{p+1}} \cdot \sum_{k=0}^{p} 2^{\mathbf{p+1}-k+1} + \ldots + E_{\mathbf{m\text{-}1}} \cdot \sum_{k=0}^{p} 2^{\mathbf{m\text{-}1}-k+1} \Bigg) - \Bigg((\mathbf{2} \cdot \mathbf{2}) \cdot E_{p+1} \\
&\quad + (\mathbf{4} \cdot \mathbf{2} + \mathbf{2} \cdot \mathbf{4}) \cdot E_{p+2} + \ldots + E_{m-1} \cdot \sum_{k=p+1}^{m-1} 2^{(m-k)} \cdot 2^{(k-p)} \Bigg) + \mathbb{L}_{\mathrm{SN}} + K'. \\
&= \mathbb{L}^{\mathrm{SP}}_{\mathrm{SC}} + \mathbb{L}^{\mathrm{SN}}_{\mathrm{SCL}} + K' - \underbrace{\sum_{m=0}^{p} \left(\sum_{j=0}^{m} E_m \cdot 2^{m-j+1} \right)}_{\mathbb{L}_1} - \underbrace{\sum_{m=p+1}^{\log_2 M-1} \left(\sum_{j=0}^{p} E_m \cdot 2^{m-j+1} \right)}_{\mathbb{L}_2} \\
&\quad - \underbrace{\sum_{m=p+1}^{\log_2 M-1} \left(\sum_{j=p+1}^{m} E_m \cdot 2^{m-j+1} \cdot 2^{j-p} \right)}_{\mathbb{L}_3}. \\
&= \mathbb{L}^{\mathrm{SP}}_{\mathrm{SC}} + \mathbb{L}^{\mathrm{SN}}_{\mathrm{SCL}} + K' - \sum_{m=0}^{p} E_m \cdot \left(2 \cdot \mathbf{2^{(m+1)}} - 2 \right) \\
&\quad - \sum_{m=p+1}^{\log_2 M-1} E_m \cdot \left(2 \cdot \mathbf{2^{(m+1)}} + \frac{\mathbf{2^{(m+1)}}}{P} \log \left(\frac{\mathbf{2^{(m+1)}}}{4P} \right) \right),
\end{aligned}
\tag{3.2}
$$

where $\mathbb{L}^{\mathrm{SP}}_{\mathrm{SC}}$ and $\mathbb{L}^{\mathrm{SN}}_{\mathrm{SCL}}$ refer to the latency of the SC semi-parallel decoder (3.1) and the

latency required to decode the constituent codes (special nodes), respectively. Also, $\mathbb{L}_1$ is the reduced latency due to constituent codes $\{E_0, E_1, \ldots, E_p\}$ of length smaller or equal to $2^{p+1} = 2P$ while $\mathbb{L}_2$ and $\mathbb{L}_3$ represent the reduced latency due to the presence of constituent codes $\{E_{p+1}, \ldots, E_{\log_2 M-1}\}$ of lengths greater than $2P$.

3.3.1 Influence of N and the number of PE on latency

In the case of semi-parallel architectures, algorithmic and architectural parameters are likely to have a strong impact on the latency when they are considered during the design of polar decoders. In order to study the influence of N and the number of PE P on the decoding latency, we plot the number of clock cycles required to decode one frame of the 5G NR polar code of four different lengths $N = \{64, 128, 512, 1024\}$ with P varying from 2 to 64.

For each value of P, three different algorithms are considered and consist of SCL, SSCL-SPC and the Fast-SSCL-SPC. Based on the speed optimization proposed for the latter [43], three additional values of $\{S_{\text{R1}}, S_{\text{SPC}}\} = \{1, 2\}, \{1, 4\}, \{2, 4\}$ are considered in addition to the optimal variant of this algorithm $\{S_{\text{R1}}, S_{\text{SPC}}\} = \{L - 1, L\}$. We refer to them as Fast-SSCL-SPC-12, Fast-SSCL-SPC-14 and Fast-SSCL-SPC-24. S_{R1} is the number of path splits in a R1 node and S_{SPC} is the number of path splits in a SPC node. Simulation results are reported in Fig. 3.8 while limiting the length of constituent codes to $M = 16$.

As expected, the decoding latency decreases as P increases. However the reduction rate of latency is not linear with respect to P as observed in (3.1) and this is due to the fact that the degree of parallelism offered by the SC decoder is reduced by half from one higher decoding stage to another lower one. This can be seen in Fig. 3.8a and Fig. 3.8d when Fast-SSCL-SPC-12 is used where the latency is reduced by 70% and 52% when P varies from 2 to 8, respectively, while it is reduced by a smaller ratio of 58% and 13% when P varies from 8 to 64, respectively. In addition to that, the same latency is obtained with $P = 64$ and $P = 32$ when $N = 64$ since a maximum number of 32 parallel operations are available to be processed in parallel.

Furthermore, we can clearly observe the impact of the algorithmic choice on the decoding latency. Therefore, using SSCL-SPC instead of SCL with $P = 8$ leads to 44%, 50%, 52% and 49% reduction in latency when N =1024, 512, 128 and 64, respectively. However, a less significant reduction in latency is measured when using Fast-SSCL-SPC instead of SSCL-SPC with $P = 8$. The latency reduction in this case is equal to 22%,

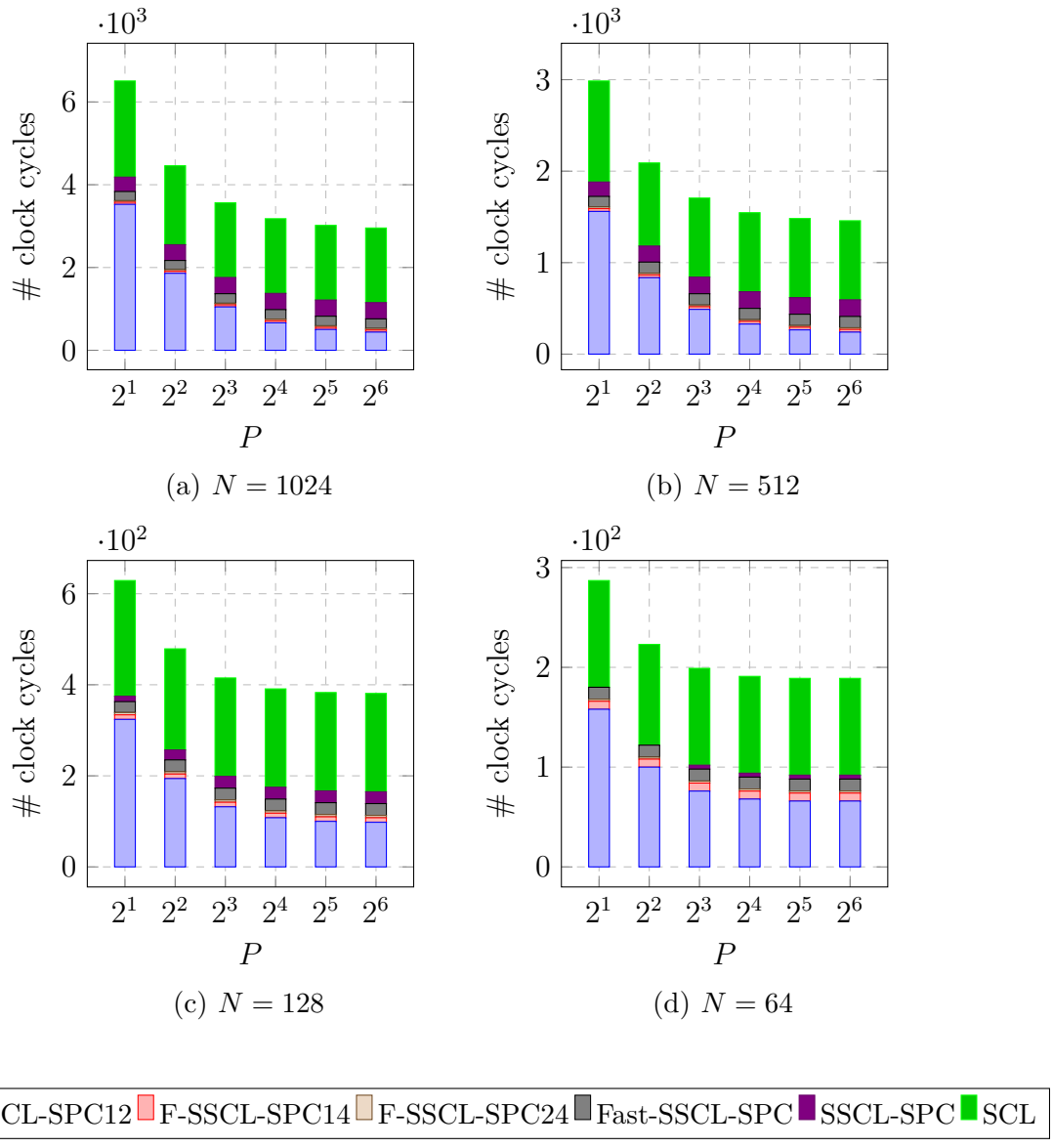


Figure 3.8 – Number of clock cycles required to decode one polar code frame for a varying number of PEs. Worst-case latency is reported while varying the value of R. Results are given for SCL and five related variants of simplified algorithms.

22%, 13% and 4% when N =1024, 512, 128 and 64, respectively. That disparity in latency between the different values of N is due to the fact that for short polar codes the presence of constituent codes is less significant in comparison to relatively long codes. In addition to that, when they are identified, their size is usually smaller than $M = 16$.

3.3.2 Influence of tree-pruning on latency

The decoding tree of polar codes may feature multiple constituent codes of different types and sizes. However, some of these identifiable constituent codes are most likely to appear at low code rates such as R0 and REP while others are most likely to appear at high code rates such as R1 and SPC. Supporting a wide range of code rates, as required in 5G NR, does not offer much flexibility in this regard. On the other hand, large constituent codes are increasingly encountered as the code length increases. To show the impact of M on latency, we plot in Fig. 3.9 the number of clock cycles required to decode one frame of polar codes of lengths $N = \{64, 128, 256, 512, 1024\}$ for $M = \{4, 8, 16, 32\}$. The same analysis is repeated with SSCL, SSCL-SPC, Fast-SSCL and Fast-SSCL-SPC.

As expected, the number of clock cycles required to decode a constituent code varies according to its type and size. Furthermore, the identification of constituent codes highly depends on M. Some polar codes may benefit better from increasing M than others, especially when they feature large low-latency decoding constituent codes. Indeed, among the set of considered polar codes, the one which achieves the best latency reduction for a fixed value of M does not necessarily produce the same achievement with M', $(M \neq M')$. This means that relying on the worst-case latency to evaluate the reduction brought by varying M on a set of rate-variable polar codes that share the same code length leads to an unfair comparison. Therefore, we consider in this analysis the average latency reduction that is obtained as M is varied. We can see from Fig. 3.9 that a decoder that can decode constituent codes of size $M = 32$ when targeting polar codes of length $N = 1024$ reduces the latency by 24%, 24%, 36% and 42% in comparison with $M = 4$ under SSCL, SSCL-SPC, Fast-SSCL and Fast-SSCL-SPC, respectively. However, a very small improvement in latency is noticed beyond $M = 32$ and is not worth to be considered. Moreover, we notice that setting M to 32 for $N = 64$ does not bring any improvement in latency since most of the polar codes at this length do not feature constituent codes larger than $M = 4$.

On the other hand, the influence of tree-pruning on latency also depends on the type of special nodes. As a result of using pruning techniques, the reduction in latency obtained

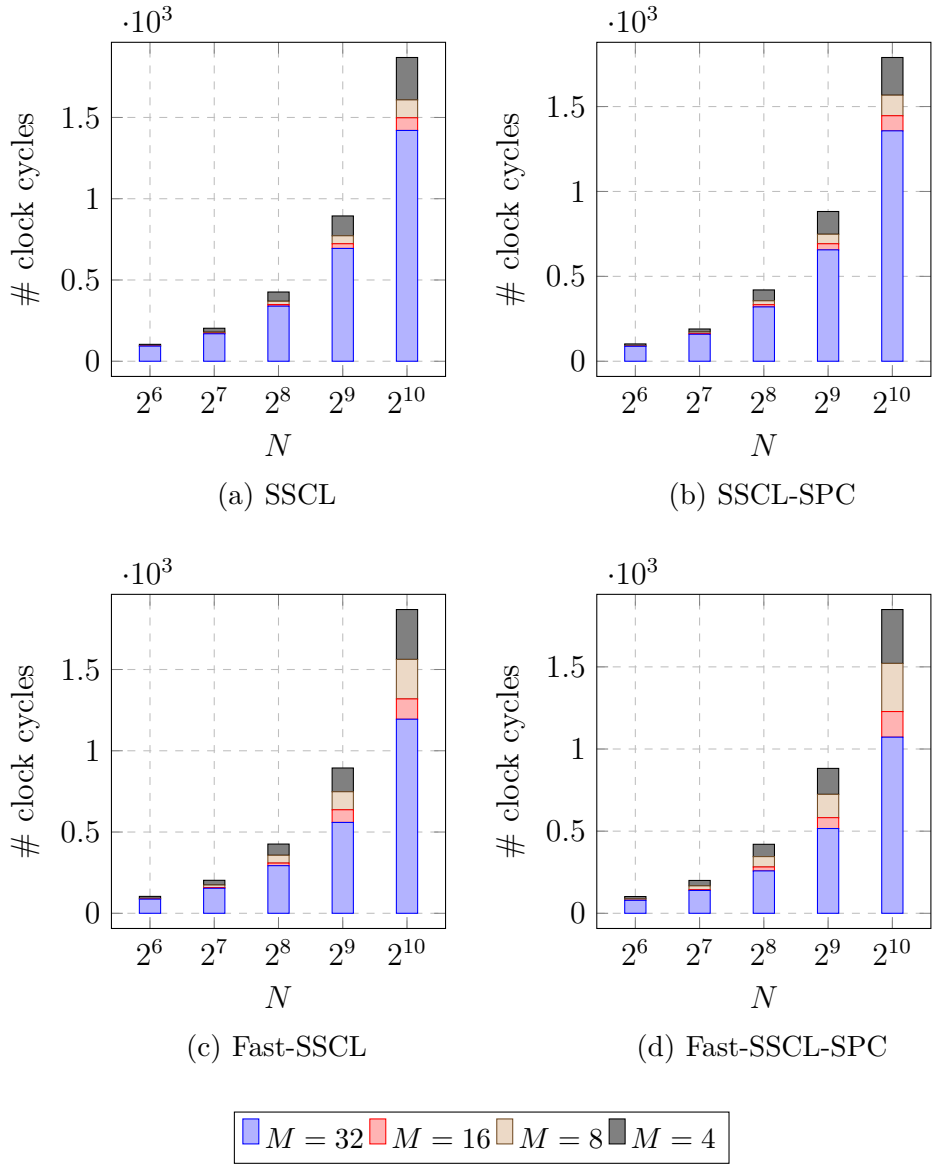


Figure 3.9 – Number of clock cycles required to decode one polar code frame as N varies from 64 to 1024. Average latency is reported while varying the value of R. Results are provided for various values of M and are reported for four different algorithms.

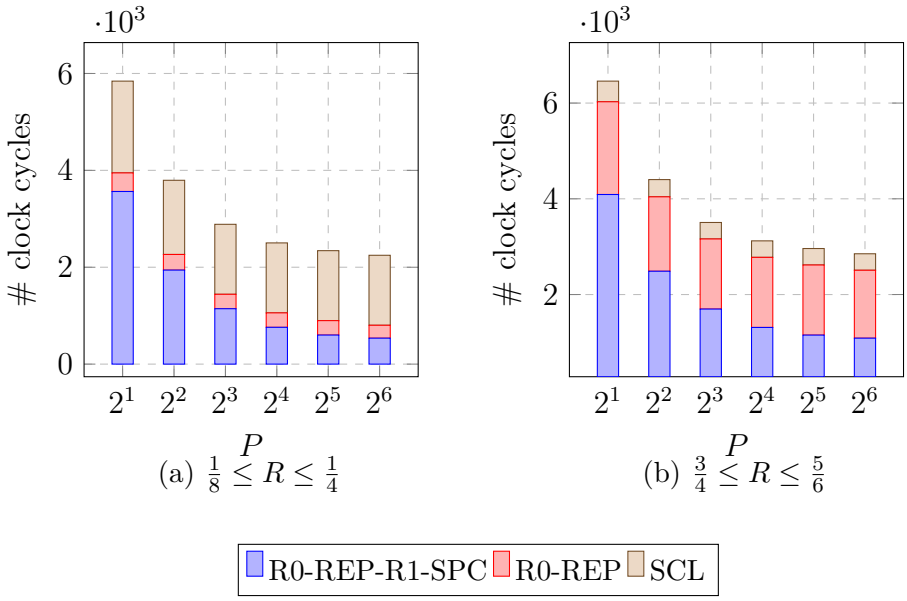


Figure 3.10 – Number of clock cycles required to decode one polar code frame for a varying number of PEs and different pruning techniques. Average latency is reported and results are given for low and high code lengths.

in Fig. 3.8 and Fig. 3.9 does not come from the same special nodes. In fact, the latency of decoding polar codes increases as the rate R increases. This observation is true for the SCL algorithm, which requires more path selection operations. Nevertheless, it is also true for simplified SCL algorithms, which feature more R1 and SPC nodes as R increases; hence several clock cycles are required to decode them. Following this observation, The considered worst-case latency reported in Fig. 3.8 is, in fact, the latency obtained from a polar code whose code rate is close enough or equal to the maximum value of $R \in [\frac{1}{8}, \frac{5}{6}]$. Therefore, this latency reduction comes directly from using R1 and SPC nodes and does not underline the impact of using R0 and REP nodes in reducing the decoding latency. Similarly, the average latency reported in Fig. 3.9 does not either provide information on the impact of special node types, except the sizes, on latency reduction. Indeed, special node types have a varying impact in reducing the average latency of several polar codes of different code rates. The different types of special nodes do not show an impact in latency reduction with the same proportion when R is varied. Therefore, to underline the impact of special node types in reducing latency, we show in Fig. 3.10 the average number of clock cycles required to decode one polar code frame at low code rates, $1/8 \leq R \leq 1/4$, and high code rates $3/4 \leq R \leq 5/6$, distinctively. For this analysis we set M to eight.

As expected, the impact of R0 and REP nodes in reducing the decoding latency is

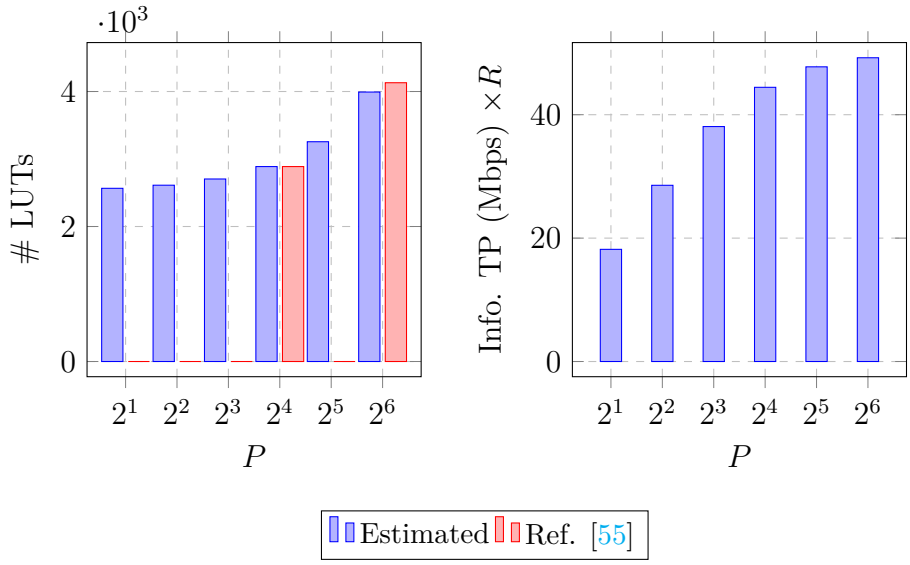


Figure 3.11 – Hardware complexity and information throughput as function of P for $N = 1024$. Operating frequency is set to 100 MHz.

more significant at low code rates. Only a few information bits are present within the frozen set, which leaves space for numerous and large low rate constituent codes, i.e., R0 and REP nodes, to be identified. Therefore, these two special nodes significantly reduce the latency of the conventional SCL algorithm from 30% to 58% for P varying from 2 to 64. However, the impact of these special nodes is less significant at high code rates. Latency is reduced by only 7% to 12% for P varying from 2 to 64. On the other hand, the impact of R1 and SPC nodes in reducing the decoding latency is more significant at high code rates where the polar code features numerous and large special nodes of type R1 and SPC. The latency reduction brought to the decoder by further adding R1 and SPC nodes to the previous SCL algorithm, which already includes the decoding of R0 and REP nodes, is minor for the polar codes of low code rates. At the same time, it is significant for the polar codes of high code rates. It can be observed from Fig. 3.10 that only 10% to 28% reduction is reported at low code rates against 32% to 57% at high code rates, for P varying from 2 to 64.

3.4 Hardware complexity and throughput analysis

3.4.1 Influence of the number of PE on hardware complexity

As the latency is decreased with increasing P, the complexity of the decoder increases. In order to analyse the relation between the complexity of the semi-parallel architecture and P, we propose to implement the processing element of [55] and estimate the complexity considering the published results for $N = 1024$ as reference. In [55], two semi-parallel architectures are designed with $P = 16$ and $P = 64$. Taking as reference the design with $P = 16$, and the logic synthesis results obtained from the design of a single PE, we estimated the hardware complexity for $P = 2$, 8, 32, and 64. In this estimation, we simply add and subtract from the reference design the hardware resources (lookup tables and flip-flops) corresponding to the number of PEs, while assuming the remaining components of the decoder unchanged. Complexity results in terms of look-up tables (LUTs), which are predominant in quantity and variation compared to flip-flops (FFs), are reported in Fig. 3.11. Comparing the results for $P = 64$ reveals a slight inaccuracy in the complexity estimation approach, which is expected as part of the design was assumed unchanged. Nevertheless, the relative comparison with respect to different values of P provides good insights on their impact on the hardware complexity. The results show that the impact of the number of PE on complexity is relatively limited. In fact, when P increases by a factor of $\times 32$, from 2 to 64, the overall number of LUTs of the semi-parallel decoder architecture increases by only a factor of $\times 1.55$. On the other hand, this increase in the number of PEs leads to an increase in information throughput by a factor of $\times 2.7$ when considering the latency expression provided in (3.1) and an operating frequency of 100 MHz. It is interesting to notice here that the increase in the information throughput when varying P gradually from 2 to 64 is not linear. It is higher for small values of P. For instance, the throughput increases by 57% when P increases from 2 to 4, whereas it increases by only by 3% when P increases from 32 to 64. This is due to the parallelism bottleneck of SC algorithms. Indeed, as the number of PEs increases, less stages of the decoding tree of polar codes can benefit from the added PEs. In the example of Fig. 3.11 for $N = 1024$, when $P = 64$ only the highest four stages make a full use of the 64 PEs. However, no change in decoding speed happens for the lowest seven stages when increasing P from 32 to 64. Extended analysis on throughput is provided in Section 3.4.3.

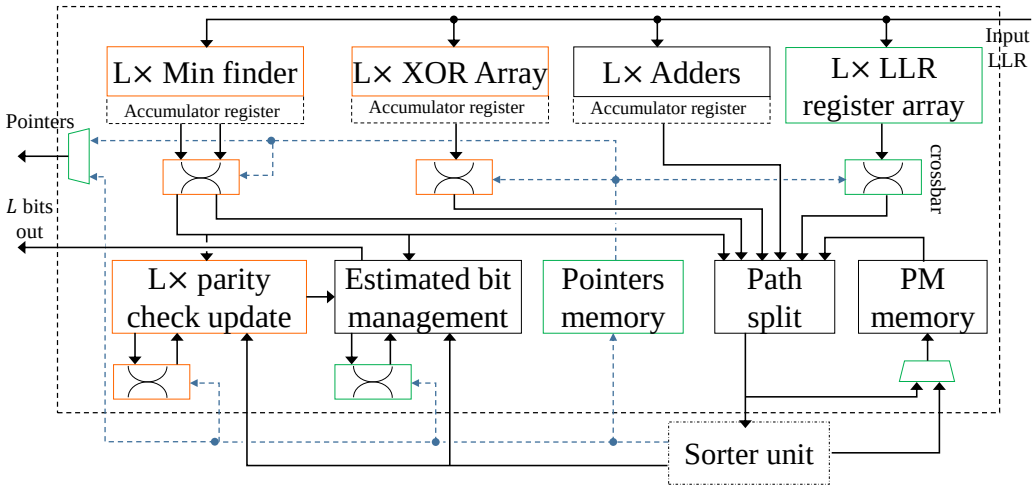


Figure 3.12 – The architecture of the SNLD designed to decode special nodes.

3.4.2 Influence of tree pruning on hardware complexity

The decoding of constituent codes obtained with tree-pruning techniques requires dedicated hardware resources, beyond those required by the classical SCL semi-parallel decoder. A specific hardware unit namely Special Node List Decoder (SNLD), depicted in Fig. 3.12, is designed to support decoding the constituent codes R0, REP, R1 and SPC according to SSCL and SSCL-SPC algorithms independently from the PEs. The complexity due to tree-pruning techniques is then evaluated taking into consideration the implementation of the SNLD for $L = 8$.

To do that, the proposed SNLD unit has been described in VHDL and synthesized on a Xilinx Virtex-7 XC7VX-485T FPGA device for three different scenarios. In the first scenario, the SNLD is designed to support only the decoding of R0 and REP nodes. The support of R1 nodes is added in the second scenario through the addition of the green-colored blocks in Fig. 3.12. The third scenario supports in addition R1 and SPC nodes through the addition of the orange-colored blocks (Fig. 3.12). SNLD allows resource sharing of the operations that are common to the different special nodes. In the synthesis results, the contribution of the sorter unit has been removed as it is unchanged for all the explored decoding algorithms. The LLR values are represented in SM format, and both LLR and PM values are quantized to 7 bits. For each of the three scenarios, M is varied from 2 to 32 assuming each time that $P = M$ so that the SNLD receives its LLRs in one clock cycle. From the synthesis results presented in Fig. 3.13, we note that the number of FFs used to decode R0-REP special nodes is almost constant regardless of M. However,

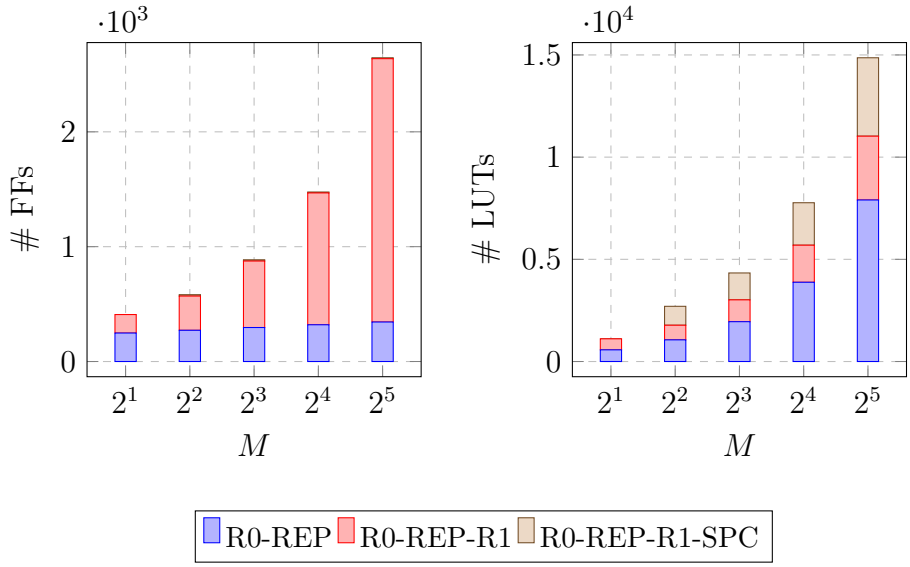


Figure 3.13 – Number of FFs and LUTs required by SNLD to decode special nodes R0, REP, R1 and SPC as a function of M according to three scenarios.

the number of FFs starts to increase with M when decoding R1 nodes. This is due to the register array, implemented to store the input LLRs until they are all processed one by one. Since the SC-based decoder does not process multiple nodes simultaneously due to its sequential nature, R1 and SPC nodes do not overlap and the same LLR register array is used store the LLRs of both nodes. Therefore no additional FFs are required during the third scenario that includes further the decoding of SPC nodes. On the other hand, the number of LUTs involved in the first scenario increases linearly with M. This is due to the fact that $M-1$ adders, designed as a tree-structure fully parallel adder, are implemented to decode R0 and REP nodes. Furthermore, the number of LUTs is increased when R1 nodes start to be considered for $M=2$ and keep increasing as M increases. This is due to the use of crossbars required to support candidate competition during the successive decoding of the bits of R1 nodes. The decoding of SPC nodes implies performing a parity check equation via a XOR array and searching for a minimum LLR value. Yet, similarly to the first scenario, the complexity of these operations grows linearly with M.

The number of LUTs used by the R0-REP-R1-SPC decoder for $M=8$ is 3.9 times larger than that for $M=2$, and that for $M=32$ is 3.43 times larger than that for $M=8$. This significant increase in the number of LUTs is mainly due to the growth in the number of used adders and comparators in the minimum finder unit.

In a second approach, we consider an accumulator register based implementation

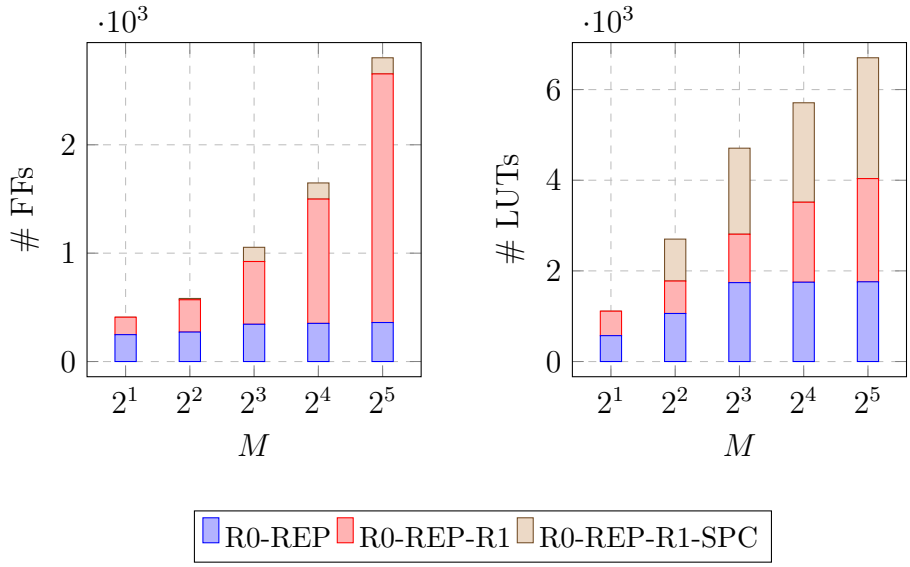


Figure 3.14 – Number of FFs and LUTs required by SNLD to decode special nodes R0, REP, R1 and SPC when considering the second approach with accumulator registers to reduce the number of adders, comparators and XOR gates.

(dotted blocks in Fig. 3.12) with reduced tree size for the adders, comparators and XOR gates. This leads to a semi-parallel architecture of the SNLD with D adders, comparators, and XOR gates, where $D < M - 1$. Synthesis results when using this second approach are shown in Fig. 3.14 for $D = 8$. When compared to the results of Fig. 3.13 for the third scenario, a significant improvement can be observed where the number of LUTs is reduced by 26% and 55% for $M = 16$ and $M = 32$, respectively. This comes at the cost of a slight increase in the number of FFs due to the presence of accumulator registers. With this approach, the increase in the number of LUTs when moving from $M = 8$ to $M = 32$ drops from $\times 3.42$ to $\times 1.42$. This significantly reduces the influence of tree pruning on the hardware complexity.

3.4.3 Influence of PE and tree pruning on throughput

Due to data dependency between nodes of the SC decoding tree, only a subset of nodes can be activated at a time. Hence, the parallelism level offered by the SC algorithm is limited. The number of operations allowed to be performed in parallel is equal to $N/2$ in the first decoding stage. However, it is continuously halved as we evolve towards leaf nodes, where only a single operation can be performed by visiting one node.

Pruning the decoder tree of SC algorithms using direct decoding of specific constituent

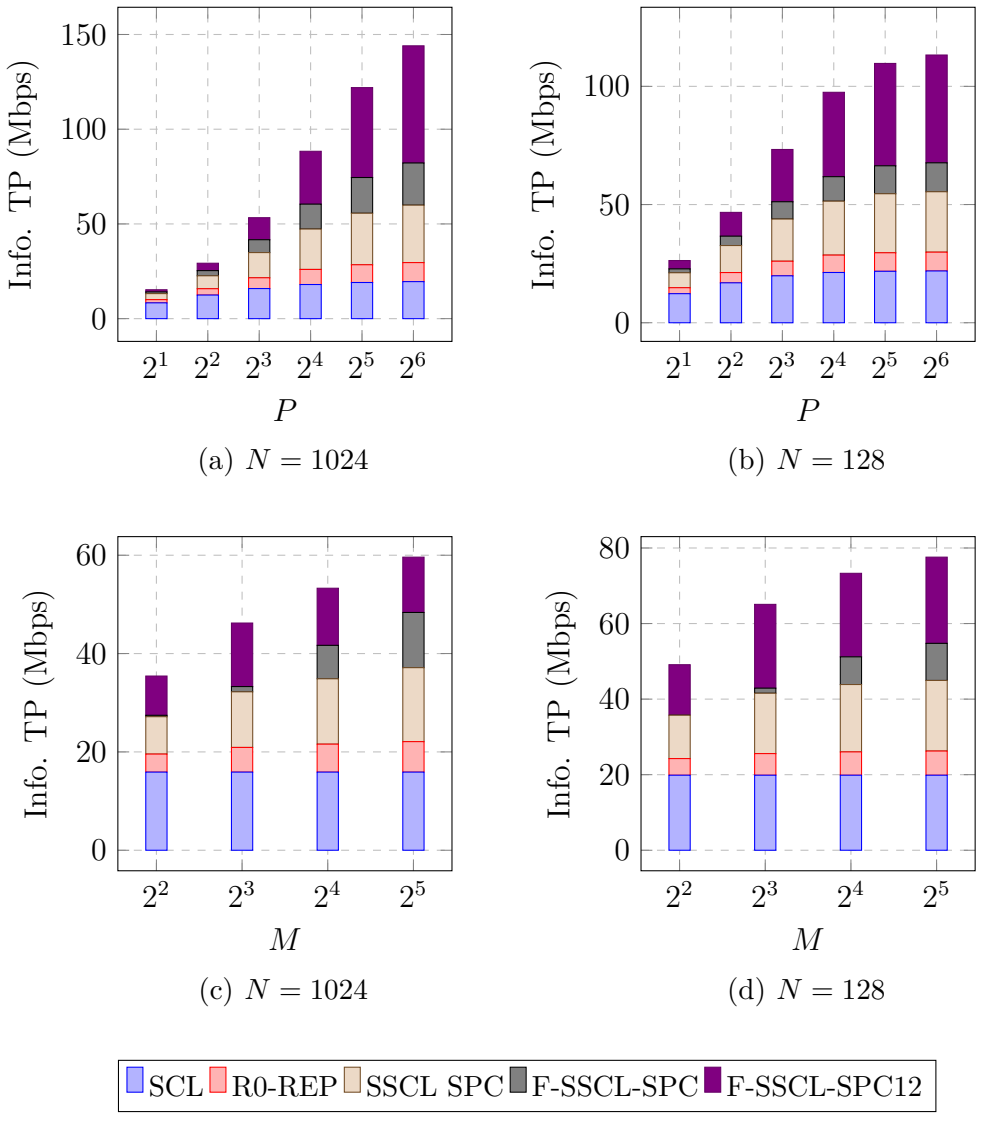
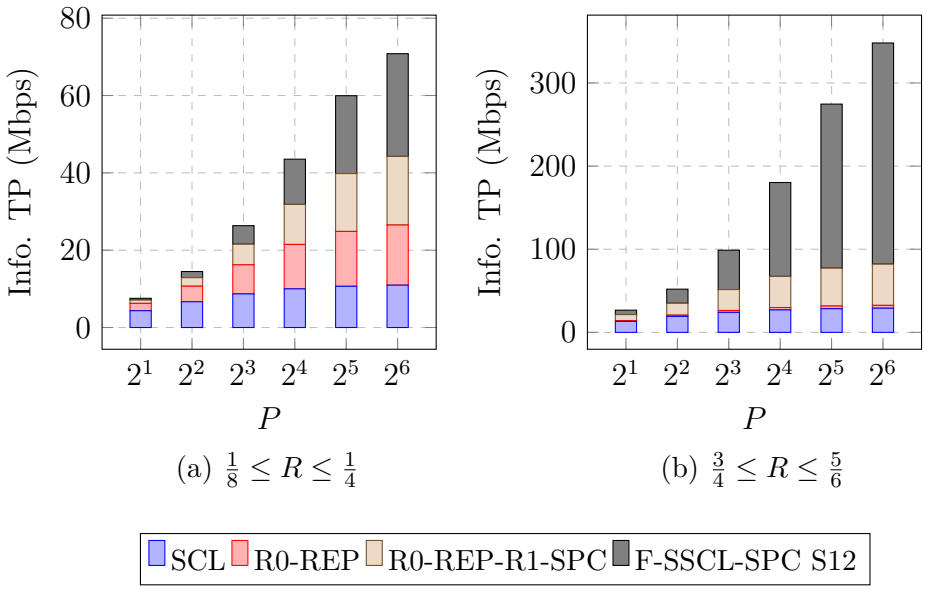


Figure 3.15 – Throughput comparison between different decoder types for different values of P and M. Two code lengths are considered $N = 1024$ and $N = 128$.

codes at earlier stages reduces the number of nodes to visit. Also, this increases the exploitable parallelism degrees of SC-based decoders since parallel decoding of bits of a special node overcomes data dependency. In this context, we propose to analyze the information throughput of 5G polar codes under the SCL algorithm and fast decoding algorithms, used so far, as a function of P then as a function of M. Throughput is calculated analytically based on the latency equation and assuming an operating frequency of 100 MHz. First, we plot in Fig. 3.15a and Fig. 3.15b the average throughput of decoding polar codes of lengths $N = 1024$ and $N = 128$ with $1/8 \leq R \leq 5/6$. The maximum length M of special nodes is set to 16. As discussed earlier, the throughput of SCL decoder increases as P increases. Nevertheless, this increase is not linear when varying P gradually from 2 to 64. Indeed, the throughput of polar codes of lengths $N = 1024$ and $N = 128$ increases by 49% and 37%, respectively, when P increases from 2 to 4, whereas it increases by only 2% and less than 1%, respectively, when P increases from 32 to 64. Furthermore, we can observe from these figures the significant addition to the throughput as soon as fast decoding algorithms relying on tree-pruning are used. For instance, the throughput of the SSCL-SPC decoder, with $P = 8$, is 2.2 times larger than the throughput of the SCL decoder for $N = 1024$. Moreover, It is 3 times higher with a larger P ($P = 64$). By tolerating a loss of 0.1 dB in error correction performance for $N = 1024$, the average achievable throughput with Fast-SSCL-1 ($S_{\mathrm{R1}} = 1$) is 144 Mbps, 7.3 times larger than with SCL.

Moreover, decoding multiple bits in parallel overcomes the drawbacks related to the limited exploitable parallelism levels of SCL algorithm. Indeed, fewer nodes are visited to compute LLRs, which reduces the number of access to the stages that do not benefit from increasing P, especially when large and various types of special nodes are considered. Consequently, the increase in throughput tends to be more linear with respect to P compared to SCL.

In order to evaluate the impact of decoding large special nodes on the information throughput, we propose to vary M from 4 to 32 and set P to 8 while using different decoding algorithms. The average throughput of decoding polar codes of lengths $N = 1024$ and $N = 128$ with $1/8 \leq R \leq 5/6$ are reported in Fig. 3.15c and Fig. 3.15d. Thus, varying M from 4 to 8 improves the average throughput by 19% and 16% for $N = 1024$ and$N = 128$, respectively. However, varying it from 16 to 32 improves the throughput by only 6% and 2% for $N = 1024$ and $N = 128$, respectively. Furthermore, one can notice that increasing M from 4 to 8 barely improves the throughput of the Fast-SSCL decoder,

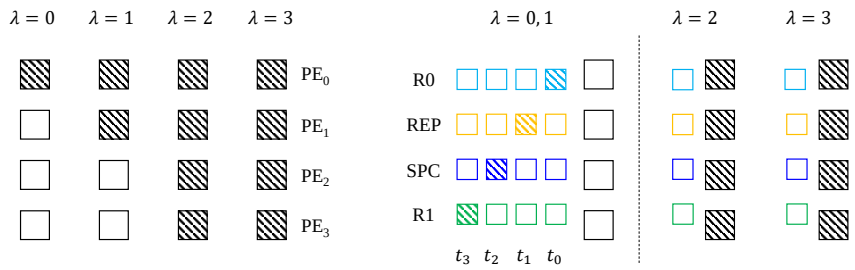


(a) $\frac{1}{8} \leq R \leq \frac{1}{4}$ (b) $\frac{3}{4} \leq R \leq \frac{5}{6}$

Figure 3.16 – Maximum information throughput of SCL decoder and various polar code decoders with different pruning techniques as a function of P. Low and high code rates are considered with $N = 1024$.

contrary to when M increases from 8 to 16. In fact, the added value of the Fast-SSCL algorithm compared to SSCL occurs on special nodes of size greater than L. However, this condition is not satisfied in our context, i.e., $L = 8$. Therefore, the Fast-SSCL decoder can enhance the throughput of the SSCL decoder starting from $M = 16$ only.

On the other hand, as shown with latency analysis, the influence of the algorithms based on pruning the decoder tree of polar codes on the throughput depends on the type of special nodes used by the decoder. The large range of rates used in the previous simulation does not reveal the importance of each special node type on improving the throughput. To analyze this effect, we present in Fig. 3.16a and Fig. 3.16b the throughput obtained for decoding the polar codes of lengths $N = 1024$ and $N = 128$ at low code rates, $1/8 \leq R \leq 1/4$, and at high code rates $3/4 \leq R \leq 5/6$, respectively. As expected, the impact of R0 and REP special nodes is more important at low code rates, while it is almost negligible at higher code rates. For instance, these two special nodes increase the throughput of the SCL decoder by 2.15 times and 2.42 times when R is low for $P = 16$ and $P = 64$, respectively. However, they increase the throughput by only 1.03 times and 1.11 times when R is high for $P = 16$ and $P = 64$, respectively. Now, when R1 and SPC nodes are considered in addition to R0 and REP nodes, a significant throughput enhancement is observed for high code rates. Indeed, R1 and SPC nodes increase throughput by 2.25



(a) SC decoder (b) SCL pruned decoder

Figure 3.17 – Number and type of active (hashed squares) and inactive (blank squares) PEs ($P = 4$) and special node decoders at each stage activation of PC(16,8) of Fig. 2.9.

times and 2.53 times for $P = 16$ and $P = 64$ when SSCL-SPC decoding is considered and by 6 times and 10.72 times when Fast-SSCL-SPC and $\{S_{\text{R1}}, S_{\text{SPC}}\} = \{1, 2\}$. Contrary to R0 and REP nodes which only bring minor throughput improvements at high code rates, the consideration of R1 and SPC nodes still enhances the throughput of the decoder at low code rates. Certainly, this increase is less significant compared to the one observed at high code rates, but quite noticeable considering that the throughput of the SCL decoder is improved by 2.67 times when Fast-SSCL-SPC and $\{S_{\text{R1}}, S_{\text{SPC}}\} = \{1, 2\}$ are considered with 64 PEs.

3.5 Hardware efficiency analysis

The strong data dependency of the SC algorithm limits the performance of the SC decoder in terms of latency and throughput. The solution, based on increasing the number of implemented PEs, becomes less efficient when this number is relatively high compared to the length of the polar code. In addition, the use of a large number of PEs decreases the efficiency of the SC decoders in terms of resource utilization rate. On another note, decoding a group of bits in parallel using tree-pruning algorithms breaks the data dependency of the SC decoder and enhances its performance. To summarize, adding further resources to the conventional decoder does not systematically improve its hardware efficiency.

3.5.1 Activity of SC decoders

To characterize the hardware efficiency of the semi-parallel SC decoder, we propose to analyze the activity of the processing units. We count the number of active processing

units (PE, special node decoders) during the time periods spent decoding one polar code frame. Fig. 3.17a shows the number of active (hashed squares) and inactive (blank squares) PEs at each stage required for the decoding of the PC(16,8) represented by the SC decoder tree of Fig. 2.9a. The number of PEs used for this example is four. We can notice that all of the implemented PEs are used during the activation of stages $\lambda = 3$ and $\lambda = 2$. However, only half and a quarter of the number of implemented PEs are used during the activation of stages $\lambda = 1$ and $\lambda = 0$, respectively. Given that a stage λ is activated $2^{n-\lambda}$ times, the number of times the semi-parallel SC decoder fully benefits from the PEs when decoding the PC(16,8), considering the additional clock cycles required to update nodes of stages $\lambda = 3$, is equal to $2 \times 2 + 4 \times 1 = 8$. On the other hand, 2 and only 1 of the 4 PEs are used 8 and 16 times respectively at stages $\lambda = 1$ and $\lambda = 0$. This means that the activation of each PE occurs $8 + 8 \times \frac{1}{2} + 16 \times \frac{1}{4} = 16$ times. Therefore, for this example, the activity γ_P of the available PEs is equal to $16 \times P = 64$. For any $p = \log_2 P$ and $n = \log_2 N$, such as $(0 \leq p < n)$, the value of γ_P is expressed as:

$$\gamma_P = \sum_{\lambda=0}^{p} 2^{n-\lambda} \frac{2P}{2^{p-\lambda}} + \sum_{\lambda=p+1}^{n-1} 2^{n-\lambda} 2^{\lambda-p} P. \tag{3.3}$$

On the contrary, two and three PEs are left unused 8 and 16 times respectively at stages $\lambda = 1$ and $\lambda = 0$ for the example of Fig. 3.17a. This means that each PE is inactive $8 \times \frac{1}{2} + 16 \times \frac{3}{4} = 16$ times. The inactivity of PEs $\bar{\gamma}_P$ is equal to $16 \times P = 64$ and is expressed as:

$$\bar{\gamma_P} = \sum_{\lambda=0}^{p} 2^{n-\lambda} \left(2 - \frac{1}{2^{p-\lambda}}\right) P + \sum_{\lambda=p+1}^{n-1} 2^{n-\lambda} 2^{\lambda-p} P. \tag{3.4}$$

To describe the hardware efficiency of the decoder, we use γ_P and $\bar{\gamma}_P$ to derive the utilization rate α. In the case of a SC semi-parallel architecture, $\alpha_{\text{SC}}^{\text{SP}}$ is expressed as:

$$\alpha_{\text{SC}}^{\text{SP}} = \frac{\gamma_P}{\gamma_P + \bar{\gamma_P}}. \tag{3.5}$$

One can notice that $\alpha_P = 1$ is only reached with one PE ($P = 1$), which means the PE is always active during the decoding process. In the above example, where $N = 16$ and $P = 4$, the utilization rate of the decoder $\alpha_P = 1/2$. Furthermore, if we assume that one PE implementing f and g operations represents twice the complexity of one PE implementing either f or g, (3.5) results in an exact reformulation of the utilization rate of the SC decoder defined in [55] as the average number of active nodes per clock cycle,

which is given by:

$$\alpha_{\mathrm{SC}}^{\mathrm{SP}} = \frac{N \log_2 N}{2P\mathbb{L}_{\mathrm{SC}}^{\mathrm{SP}}}, \tag{3.6}$$

Following this assumption, the maximum value of the utilization rate is $\alpha_P = 0.5$ for $P = 1$. This is obvious since the implemented PE is always active during decoding. However, this same PE is used alternately to execute f or g functions per node activation. We plot in Fig. 3.18a the utilization rate as a function of P of the decoders SC, SCL, SSCL, Fast-SSCL, and Fast-SSCL-1 ($S_{\mathrm{R1}} = 1$) of lengths $N = 1024$ and $N = 128$. Only the activity of the PEs is studied, while that of the special node decoder is ignored. First, we can see that for the same number of PEs, the α_P of $N = 1024$ is higher than the α_P of $N = 128$. This is because $n - p$, the number of stages where all the P PEs are used, increases with N. Then, it can be noticed that the α_P of the SCL decoder is worse than that of the SC decoder. This is due to the additional decoding clock cycles required to perform candidate competition between different codeword paths when decoding information bits. The value of α_P, in this case, depends on the code rate R. However, for $N = 1024$ and $M = 16$, the SSCL algorithm improves α_P by 1.14 times and 1.85 times for $P = 4$ and 32, respectively, compared to the SCL decoder. This improvement is more significant with the faster Fast-SSCL-1 and results in a 1.34 times and 2.85 times increase in α_P. This is due to the latency reduction acquired by pruning the decoder tree at low decoding stages where a relatively large proportion of the PEs is idle. The PC(16,8) of Fig. 2.9 is pruned at $j = 2$. Thus, nodes at stages one and zero are not visited anymore during the decoding process. Instead, special node decoders are implemented. The number of active (hashed squares) and inactive (blank squares) PEs together with the special node decoders (colored squares) for R0, REP, SPC, and R1, is shown in Fig. 3.17b. Since special nodes are decoded in different time periods, their respective decoders are duplicated four times at stages $j = 0$, 1 to show which decoder is active (hashed square) and in what order. We see that all the PEs are used whenever stages $j = 3$ and $j = 2$ are active and idle during special node decoding. In case the Fast-SSCL-SPC-12 decoder is used, the activation of each PE occurs eight times during the 14 clock cycles required to decode one frame, which is higher than the SCL decoder, where it occurs 16 times during 40 clock cycles. The decoding of R0 nodes consists in summing up to M LLR values, and the decoding of REP nodes consists in summing up to M positive LLR values and M negative LLR values, then decoding the information bit. Thus, their computational complexity is equal to M and $2M + 1$, respectively. Following this, the utilization rate of α_{R0} and α_{REP} of the example of Fig. 2.9

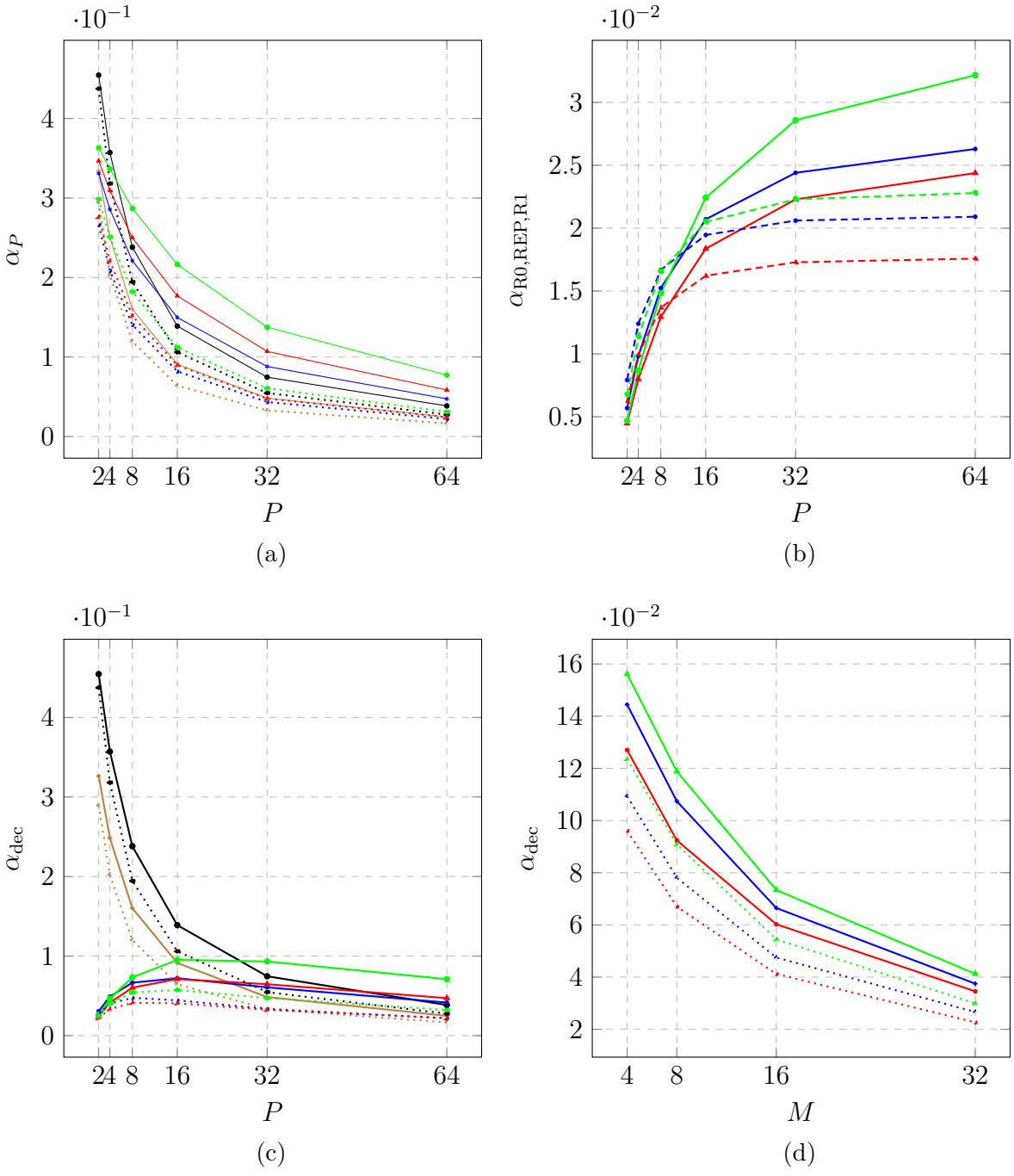


Figure 3.18 – Average utilization rates of several SC-based decoders of length $N = 1024$ (solid lines) and $N = 128$ (dotted lines): (a) $\alpha_P(P)$, (b) $\alpha_{SND}(P)$, (c) $\alpha_{dec}(P)$, (d) $\alpha_{dec}(M)$

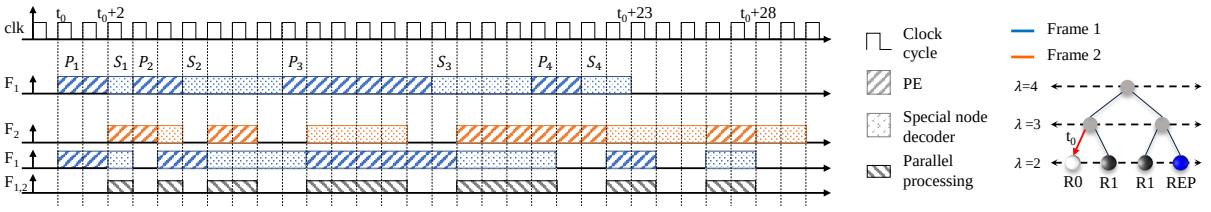


Figure 3.19 – Decoding one and two codewords of the same polar code using SSCL algorithm.

in the case of SSCL decoding is $\alpha_{\text{R0}} = M/17M$ and $\alpha_{\text{REP}} = 2M/(34M+2)$. Furthermore, the utilization rate $\alpha(Q)$ of an architecture which comprises Q processing units including PEs and various computational complexity special node decoders is expressed as a function of γ_q and $\bar{\gamma}_q$ as:

$$\alpha(Q) = \frac{\sum_{q=1}^{Q} \gamma_q}{\sum_{q=1}^{Q} \gamma_q + \bar{\gamma}_q}, \tag{3.7}$$

where γ_q and $\bar{\gamma}_q$ are the activity and the inactivity of the processing unit q $(1 \leq q \leq Q)$.

We plot in Fig. 3.18b $\alpha_{\text{R0,REP,R1}}$, the average utilization rate of the decoders SSCL, Fast-SSCL, and Fast-SSCL-1 as a function of P while setting M to 16. We consider the activity of the special node decoders of R0, REP, and R1 nodes and ignore the activity of the PEs. As expected, when P increases, the processing of nodes f and g becomes faster, and the activation rate of special node decoders during the decoding period increases, which improves $\alpha_{\text{R0,REP,R1}}$. However, this utilization rate is very low compared to α_P presented in Fig. 3.18a. This is due to the high computation complexity of the special node decoders compared to that of a PE. In addition, the PEs are more used during the decoding process than any of the implemented special node decoders. We also note that, unlike α_P, $\alpha_{\text{R0,REP,R1}}$ is better for $N = 1024$ than for $N = 128$, especially for $P \geq 16$. This is because the decoder of length $N = 1024$ benefits better from increasing P beyond $P = 16$ than the decoder of length $N = 128$. On top of that, using special node decoders of size M with the knowledge that polar codes of length $N = 128$ rarely comprise special nodes of this size leads to poor hardware efficiency. Also, we can observe that $\alpha_{\text{R0,REP,R1}}$ of the SSCL decoder is better than $\alpha_{\text{R0,REP,R1}}$ of the Fast-SSCL. In fact, since Fast-SSCL-1 searches for the minimum LLR out of M LLR values, its computational complexity is M times larger than the SSCL. Hence, its reduced latency benefit is not worth the penalty in terms of hardware efficiency. However, the $\alpha_{\text{R0,REP,R1}}$ of the Fast-SSCL-1 decoder is much better than that of the Fast-SSCL since, for the same computational complexity,

the Fast-SSCL-1 decoder is much faster. In Fig. 3.18c we plot α_{dec}, the average utilization rate of the decoders SC, SCL, SSCL, Fast-SSCL and Fast-SSCL-1 ($S_{\text{R1}} = 1$) as a function of P for $M = 16$ using (3.7). All the processing units are considered, including PEs and special node decoders of R0, REP, and R1. We can clearly see that for $P < 16$, the SCL algorithm presents a better hardware efficiency than all the simplified algorithms used for this study. When $P < 16$, the number of clock cycles to process non-pruned nodes is very large compared to the number of clock cycles to decode special nodes, i.e., special node decoders are longer idle than active. However, starting from $P = 16$ and $P = 32$ respectively, Fast-SSCL-1 and SSCL become more efficient than the SCL decoder. Using special node decoders increases the decoder throughput, reduces the latency of decoding one polar code frame but decreases the efficiency of the decoder in terms of hardware complexity. Indeed, additional computational units proportional to M are needed to compute the M data inputs related to the top constituent codes nodes. In addition to that, polar codes often include special nodes of variable lengths, which means that a special node decoder designed for M does not frequently take total usage of its hardware resources, especially when M is relatively large. This is shown in Fig. 3.18d, where the average utilization factor of SSCL, Fast-SSCL and Fast-SSCL-1 decoders is computed for $P = 8$ and various values of M for $N = 1024$ and $N = 128$. It clearly demonstrates that increasing the special node decoder size M reduces the hardware efficiency. Recall that the complexity of one PE is assumed to be twice the complexity of a processing element able to process either f or g operation in the analytical computation of utilization rates of Fig. 3.18. In addition to that, special node decoders are considered as separate hardware designs. Each decoder has a specific computational resource complexity related to the type and size of the special node. Therefore, when specific hardware optimizations such as arithmetic resource sharing are used, the utilization rate of the decoders is improved.

3.5.2 Proposed multi-frame decoding techniques

The poor hardware efficiency of SC decoders is not restricted to semi-parallel SC architectures but also to line and tree architectures [55]. Also pipelined, the tree architecture comprises $N - 1$ PEs, where 2^{j-1} PEs are instantiated at each stage j to perform the available operations whenever the stage is activated. In this way, PEs that belong to non-active stages remain inactive. After duplicating some decoding stages and by adding some PEs, a vector overlapped SC architecture that uses these stages to decode multiple received frames in parallel was presented in [56]. However, memory is also duplicated as

many times as the number of vectors decoded in parallel. As a result, this architecture can decode a maximum of $n = \log_2 N$ codewords in parallel without duplicating all the computational resources of the decoder. On the other hand, semi-parallel SC and SCL architectures do not offer many options to overlap the decoding of multiple codewords. The idea of using the idle PEs requires additional hardware complexity to maintain the routing network, but this comes at the cost of a added control complexity. Nevertheless, at the cost of added memory, a semi-parallel decoding architecture similar to the one studied in the previous sections and featuring dedicated special node decoders can be used to enhance the level of parallelism without duplicating computational resources. Precisely, an added memory for holding codeword bits is only required for a length-flexible design when the total length of codewords decoded in parallel does not exceed the designed decoder length. It was shown in Fig. 3.17 that PEs and special node decoders work alternately during the decoding process. When PEs are used, the special node decoders are not, and vice versa. Therefore, two codewords can be decoded simultaneously with the same hardware resources by alternately updating stages $j = 2, 3$ and $j = 0, 1$. This improves the throughput and hardware efficiency but also increases the decoding latency. Fig. 3.19 shows the timeline of decoding a sub-tree of one and two received frames of the same polar code using the SSCL algorithm. The polar code selected for this example comprises four consecutive constituent codes of length four, which are R0, two R1s, and REP codes. The number of PEs used by the decoder is $P = 2$. The SSCL decoder alternates between PEs and special node decoders during 23 clock cycles to decode the four special nodes of the sub-tree starting at t_0. Over this time period, PEs are activated 12 times while special node decoders are activated 11 times. The decoding process time is almost equally shared between PEs and special node decoders. This motivates the study of parallel decoding of two frames, F_1 and F_2, using the same architecture. The timeline of decoding F_1 and F_2 shows that these frames can proceed to use the available processing units, i.e., PEs and special node decoders without conflict. However, they cannot use the same type of processing units simultaneously. Therefore, F_1 starts using PEs during two clock cycles denoted by P_1 then uses the special node decoder to decode R0 for one clock cycle denoted by S_1, letting F_2 uses the freed PEs, in turn, for two clock cycles. Nevertheless, since $P_1 > S_1$, F_1 has to wait for $P_1 - S_1$ clock cycles to resume decoding. Similarly, F_2 has to wait for $P_2 - S_1$ clock cycles in order to use PEs the second time around. The remaining special nodes can be similarly decoded with the same schedule and resource sharing process. $F_{1,2}$ highlights the different time periods where both PEs and special node decoders are

activated simultaneously by the decoding process of frames F_1 and F_2 (16 clock cycles). The total amount of time required to output the codeword bits of one frame corresponds to 28 clock cycles against 23 clock cycles previously. As a consequence, latency is increased by five clock cycles compared to the case where multiple frame decoding is not applied. The total latency of decoding F frames can be expressed as:

$$\mathbb{L}_M = \sum_q (F-1)\max(P_q, S_q) + \max(P_{q+1}, S_q), \tag{3.8}$$

where P_q is the number of clock cycles required to process nodes of the pruned decoder tree before decoding the next special node, which requires S_q clock cycles. The average latency and maximum throughput of the 5G polar code over several R values when two frames are decoded in parallel is measured using (3.8) for $P = 16$ and $M = 16$. The results are provided in Fig. 3.20 for various polar code lengths. We can see that the latency is increased by 36%, 39%, 43% and 47% while throughput is increased by 20%, 21%, 24% and 28% for $N = 64, 128, 256$ and 512. This increase in latency is more significant compared to the latency analysis conducted above on the simple example of Fig. 3.19.

Although the overall decoding time is equally shared between PEs and special node decoder, it is frequently uneven between two successive activation of PEs and special node decoders, which results in relatively high decoding latency. For instance, two consecutive 16-length R1 special nodes that share the same parent node lead to increasing the decoder's latency by 15 clock cycles. However, using such a technique guarantees a latency of decoding F frames of length N/F lower than the worst-case latency of decoding one frame of polar codes of length N.

The proposed technique of decoding multiple received frames of the same polar code in parallel presented in this section is based on the notion that the different frames must alternate between using PEs and special nodes with a similar time budget during the decoding process. Therefore, with the same resource, this technique provides good hardware efficiency under the SSCL algorithm but quickly reaches its limits after only two frames are decoded in parallel. However, if faster algorithms are used, particularly the sub-optimal Fast-SSCL, i.e., $S_{\mathrm{R1}} < L - 1$, the time required to decode the set of identified special nodes is much lower than the time required to perform f and g operations.

In this context, we propose another technique that consists in duplicating the PEs as many times as the number of decoded frames in parallel. These multiple frames share the same special node decoder, which increases its utilization rate. Therefore, the decoder

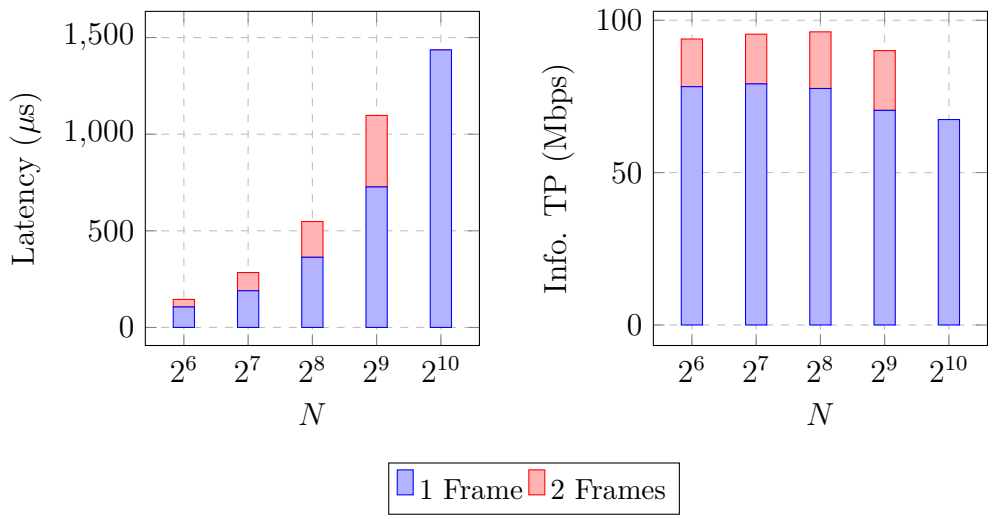


Figure 3.20 – Latency and throughput as a function of code length N for a multi-frame SSCL decoder.

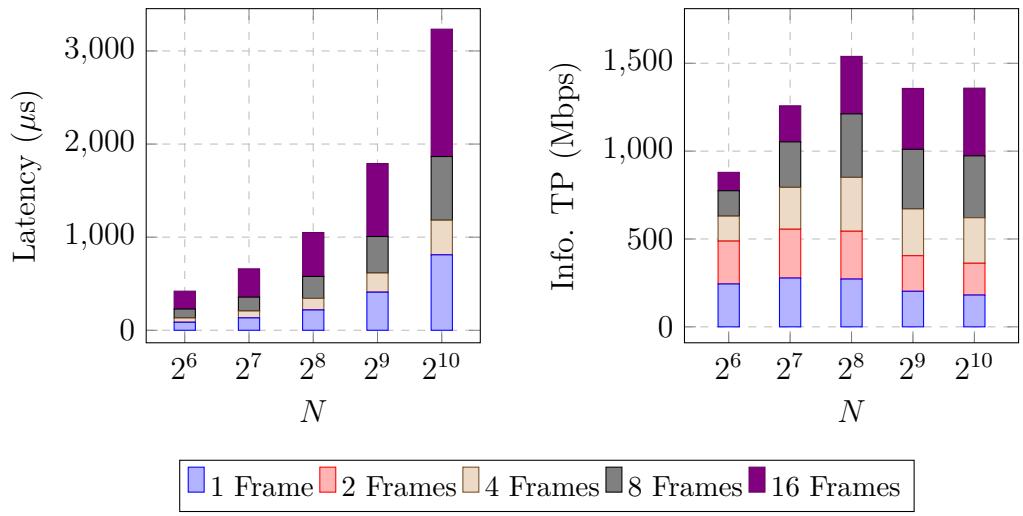


Figure 3.21 – Latency and throughput as a function of code length N for a multi-frame Fast-SSCL-1 decoder.

architecture includes $F \times P$ PEs and one special node decoder, where F is the number of frames to decode in parallel. The maximum latency and maximum throughput of the Fast-SSCL-1 decoder of lengths $N \in [64, 1024]$ when decoding $F = 1, 2, 4, 8$ and 16 received frames of the same polar code are reported in Fig. 3.21. The number of PEs and the maximum length of special nodes used in this analysis are $P = 16 \times F$ and $M = 16$, respectively. We see that the latency of decoding two frames is the same as decoding one frame, which is reflected by doubling the throughput of the decoder. A small increase in latency follows increasing F above $F = 2$. Indeed, it reaches an average of 51% and 80% on all the considered code lengths $N \in [64, 1024]$ when increasing F from two to four and eight to sixteen, respectively. This increase in latency is very low compared to the latency required to decode multiple frames when they are decoded one at a time. For instance, in the latter case, the latency of decoding $N = 128$ and $F = 8$ equals 3272 clock cycles while it is equal to 1007 clock cycles (30%) using the proposed technique. Consequently, the throughput of the decoder is significantly enhanced. For instance, it is 3.13, 4.46, and 5.66 times better when decoding four, eight, and sixteen frames of length $N = 256$ in parallel than decoding a single polar code frame. Nearly similar results are obtained for other code lengths than $N = 256$, as shown in the figure. Far from being limited to $F = 2$, this technique of decoding several received frames in parallel when using the Fast-SSCL decoder provides many advantages in terms of throughput and hardware efficiency at the expense of high memory requirement and duplicated processing elements. Finally, latency and throughput analysis results presented in this section for both multiple frame decoding techniques are performed considering frames from the same polar code, i.e., the same frozen set. Further analysis can be done by considering decoding multiple frames of a variable rate and length.

3.6 Summary

A software simulation platform dedicated for polar codes was developed and presented in this chapter. It includes the whole encoding and decoding chain, from bit-channel allocation, CRC bits attachment and interleaving schemes to rate matching as defined in the 5G NR, and their reverse operations to ensure the decoding phase process. Thanks to the completeness of the software platform, bit error rate performance of 5G NR polar codes under fast decoding algorithms was evaluated and various fixed-point decoders were compared to the floating-point ones providing by this opportunity an efficient tool to select

the best quantization scheme for hardware polar decoders.

Furthermore, we presented in this chapter a design space exploration of polar decoders and provided a detailed analysis of the impact of main code and decoder design parameters on latency, throughput, hardware complexity and hardware efficiency for polar decoding architectures when targeting the flexible 5G NR polar code. In this context, the semi-parallel architecture model proves to be the most suited thanks to a broader algorithmic and architectural flexibility at design level. Therefore, based on a detailed analytical study and logic synthesis results, the latency, throughput and complexity of the decoder were evaluated for multiple variants of fast SCL decoding algorithms and for a varying number of processing elements. Results have shown that length- and rate-flexible designs limit the benefit of increasing the number of processing elements and advocate for defining various types of constituent codes while increasing the level of tree pruning. Indeed, while the use of a large number of processing elements brings a significant latency reduction at large frame sizes, this benefit becomes negligible for short frame sizes, hence penalizing the hardware efficiency of the decoder. Furthermore, some special constituent code types are more likely to appear at low code rates, such that R0 and REP, while others are more likely to appear at high code rates, such as R1 and SPC. Adding to that, the length of these special constituent codes in number of involved bits decreases with the polar code frame size. This can significantly impact implementation efficiency metrics. Therefore, multiple trade-offs between algorithmic and architectural parameters can be drawn from these results. In this regard, we proposed two multi-frame decoding approaches increasing the throughput and improving the processing units activity at the cost of additional memory resources and latency. Finally, the analysis conducted in this chapter can be further performed on any other set of polar codes and extended to support list size variation.

Based on the results of this chapter, multiple implementation choices of polar decoder are offered while providing a large spectrum of complexity/performance compromises. In the next chapter, these results are used to propose a flexible decoder for 5G NR polar codes, taking into consideration the stringent constraint on latency while being of comparable complexity to state-of-the-art polar decoders.

Chapter 4

Flexible 5G polar decoder architecture

Contents

Motivated by the need to provide a hardware-efficient polar decoder that supports the required flexibility and latency levels for 5G NR, we propose in this chapter a novel hardware architecture targeting FPGA devices and offering several features. In particular, downlink and uplink 5G NR control channel compliance with full rate and frame size support. Thanks to a special node identifier, the proposed decoder continues to benefit from tree pruning techniques to speed-up the decoding whilst maintaining compliance with the 5G NR and the various defined combinations of code rate and code length. A special node decoder is designed to efficiently decode the specific constituent codes identified for each polar code. This chapter is organized as follows. Section 4.1 presents the top-level architecture of the proposed decoder and provides a detail description of the different elements that compose it. Section 3.2 presents the implementation results and detailed comparisons. Section 4.3 presents the implementation of multi-frame decoding architecture and latency performance analysis. Finally, Section 3.6 concludes the chapter.

4.1 Proposed decoder architecture

Based on the decoding flow of the SCL algorithms described in Sections. 2.2.2 and 2.3.3, an efficient polar decoder architecture for the NR code is designed. Hence, most of the decoder components are duplicated L times to ensure a parallel decoding of the list of L candidate codewords (paths). The top-level architecture of the decoder is depicted in Fig. 4.1. Due to the sequential nature of the SCL decoding algorithm and the high dependency between the internal LLRs and bit estimates, the LLR computation and bits estimation are performed alternately. Therefore, the overall architecture is divided into two separate parts called *Soft Data Part* (SDP) and *Hard Data Part* (HDP). The proposed architecture features the following main units:

— $L \times P$ *Processing Elements (PE)* to compute and update the LLRs of the L coexistent paths.

— *Memory resources* to meet the different storage needs of different data types of the decoder such as LLRs, PMs and codewords. The main block memories are: Channel LLRs, Internal LLR, Paths, Path Metrics and Pointer Memories in addition to some register-based local arrays that will be described shortly.

— *Crossbars* to manage candidate competition at different areas whether for copying data to some freed memory locations or to ensure a correct routing of data as a result of the use of pointer memories

— *Special Nodes List Decoder (SNLD)* to perform the arithmetic operations required to decode special nodes, path split and path selection operations and to produce the codeword bits. To do this, SNLD is designed to be in charge of decoding the different constituent code types when encountered and to perform, in the absence of the latter (i.e. single bit), the classical SCL decoding.

— *Partial Sum Network* (PSN) to produce PS used by the PEs whenever the g- function is performed. It is designed to handle multi-bit decoding techniques.

— *CRC bits calculation unit* to perform CRC check operations in the case of multi-bit decoding as soon as new bit estimates are available. This operation is performed in parallel to the decoding process.

— *Extraction bits unit* to output serially the A information bits of the final path selected on the basis of the results of CRC checks using the input de-interleaver values.

— *Control unit* including a Finite State Machine (FSM) to generate all the control signals needed for the different decoder units.

— *Special node identifier (SNI)* to search and identify special nodes on-the-fly.

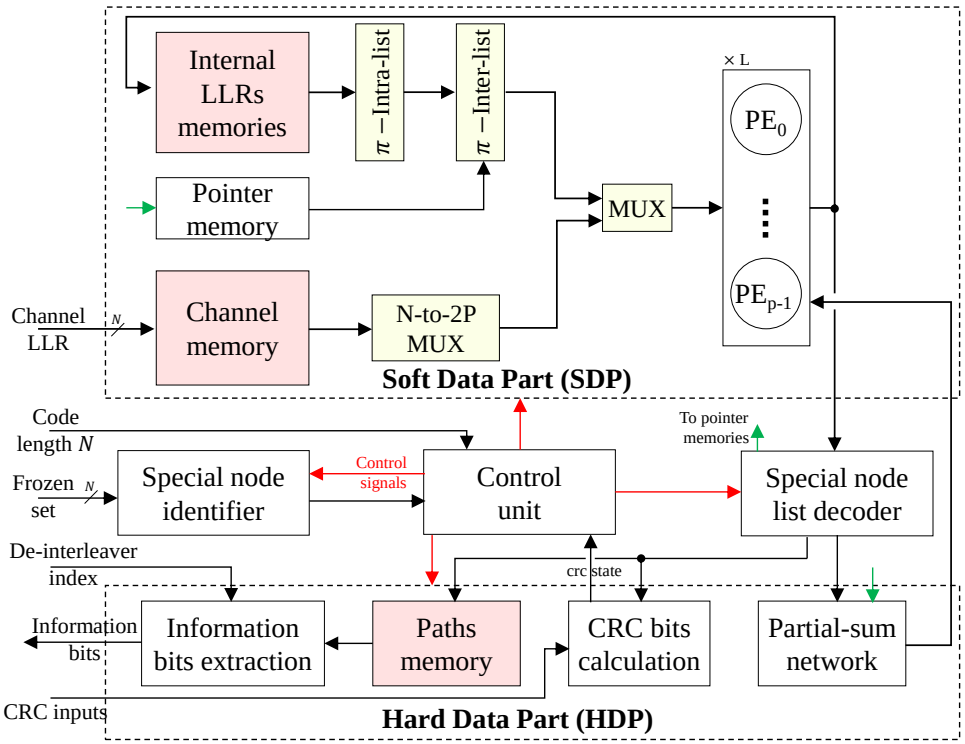


Figure 4.1 – Top-level architecture of the proposed decoder.

Referring to the SC decoding tree of Fig. 2.9.b and starting from a given node, the LLR values are produced in the SDP by a group of processing elements P and passed from parent to child nodes until either a leaf node or a special node is reached. The semi-parallel architecture of [55] in which only one stage is activated at a time is used for that purpose. LLR values are stored in a dedicated memory bank in order to be used later during the second activation of the nodes to feed the next parent's child in the decoding tree. Next, the decoding process moves to the SNLD where the different constituent-code types described in Section 2.3.2, i.e., R0, REP, SPC and R1, are decoded and path selection is performed while PEs remain inactive. The hard bit estimates including information codeword bits and partial sums are generated in the SNLD and are sent to the HDP. In the latter, the CRC bits are calculated while the decoded bits are fed back to the SDP in the form of partial sums to resume the computation of the LLRs. When decoding the PDCCH polar codes, the decoding process may stop at any time if the CRC verification fails (early termination), otherwise the decoding process continues. Besides the SDP and HDP, on-the-fly identification of special nodes and a control unit is designed in order to generate all the required control signals that the decoder needs. The different components of the decoder architecture are detailed in the following subsections.

4.1.1 Memory structure

In the tree structure of the SCL decoder, each parent node is connected to two child nodes to which it sends its LLR value in distinct time periods. Consequently, the decoding relies on dedicated memories to keep the LLR values available for the computation units as long as they are needed. For instance, the channel input LLRs are needed twice, at the very beginning of the decoding and at halfway, for the computation of the internal LLRs. Meanwhile, they are stored in a register of $N \times Q_c$ bits, where Q_c is the quantization level of the channel LLRs. However, internal LLRs are stored in a dual-port RAM-based memory configured with one write port and one read port. This choice is motivated by the ability to activate the PEs in the next clock cycle following LLR updates. In this way, one port of the dual-port RAM is dedicated to reading the previously updated LLRs and the other is used for writing the currently updated LLRs. The total number of LLR updates performed before proceeding to the lower $j-1$ stage is equal to 2^j. However, only P LLR updates are allowed to be performed at once. Thereby for stages j where $2^j > 2P$, a total of $2^j/(2P)$ time steps are needed before proceeding to the lower stage while only one time

step is needed for the other stages. The depth of this memory is thus:

$$MEM_{\text{LLR}} = \sum_{j=p+1}^{n-1} 2^j/2P + \sum_{1}^{p} 1 = \frac{N}{2P} + \log_2 P - 1, \tag{4.1}$$

while $2 \times P \times Q_{\text{i}}$ represents its width and Q_{i} is the number of quantization bits of the internal LLRs. Given that the LLRs produced by the PEs at decoding stage j are indexed from 0 to $2^j - 1$, the input LLRs to the PEs have to follow the bit-reversed indexing scheme. For example, considering the PC(16, 8) of Fig. 2.2 (Chapter 2), the decoder at stage 2 computes $L_{2,2}$ by using $L_{3,2}$ and $L_{3,6}$ and computes $L_{2,3}$ by using $L_{3,3}$ and $L_{3,7}$. Hence, the produced LLRs at a given stage by the PEs need a certain reordering before being mapped back to these same PEs to produce the LLRs of the lower stage. To avoid the need for multiplexing the LLRs at the input of the PEs, the equivalent operation is directly embodied in the control unit and two RAMs (instead of one), each of width PQ_i, named $\text{RAM}_{0,l}$ and $\text{RAM}_{1,l}$ are implemented for each path l. For higher stages, where $2^j > 2P$, LLRs are first stored in $\text{RAM}_{0,l}$ during half of the $2^j/(2P)$ time period required for computation before storing the remaining produced LLRs during the second half of the time period in $\text{RAM}_{1,l}$.

However for lower stages, all the LLRs produced during one time period are stored in $\text{RAM}_{0,l}$. In this way, the LLRs contiguity is ensured with no additional complexity. The memory structure used to store internal LLRs is illustrated in Fig. 4.2a while the component $\text{RAM}_{0,l}$ is drawn in Fig. 4.2b. The control logic for mapping the output of the processing elements PE_p to $\text{RAM}_{0,l}$ and $\text{RAM}_{1,l}$ described above is given by the following expression:

$$\text{MAP}_{\text{LLR}}^{\text{PE}\longrightarrow\text{RAM}}(t,l) = \underset{k\in 0,1}{\text{RAM}_{k_l}} \left(\sum_{q=j}^{n-1} \frac{2^q}{2P} + t \pmod{\frac{2^j}{4P}}, \text{PE}_p \right), \tag{4.2}$$

where t is the time step at which the LLRs are updated, 2^j is the number of LLRs to update at stage j. However, this mapping can no-longer keep providing the PEs with the appropriate inputs from RAM blocks when operating at lower stages. This is due to the fact that one single memory, $\text{RAM}_{0,l}$, is sufficiently large in this case to store all the LLRs relative to a single decoding stage. As a consequence, the LLR pairs used by each PE are not stored in two separate locations as planned. Therefore, an extra network of multiplexers called π-*intra-list* is implemented as shown in Fig. 4.3b for $P = 8$. It allows

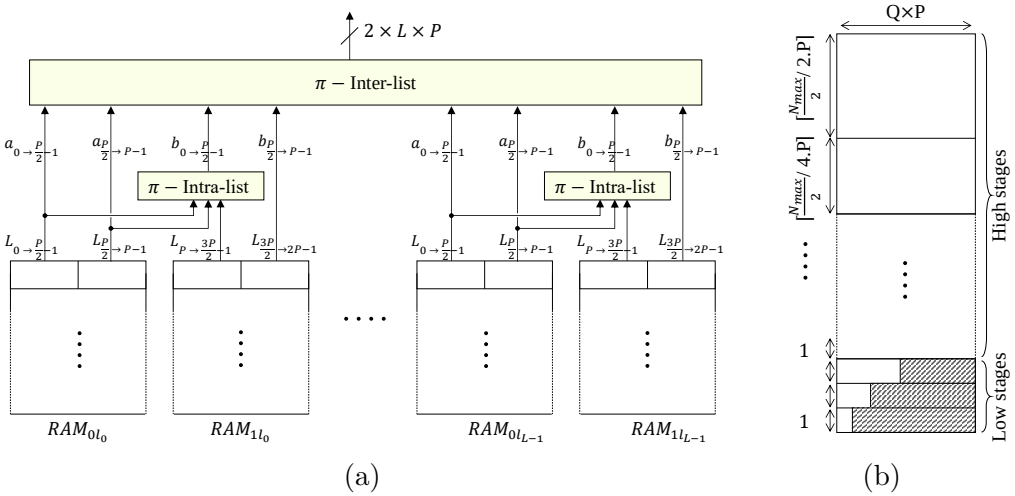


Figure 4.2 – Internal LLR memory structure: (a) RAM-based memory structure of internal LLRs and their pathway to PEs, (b) Organisation of $\mathrm{RAM}_{k,l}$.

the following mapping:

$$\mathrm{MAP}_{\mathrm{LLR}}^{\mathrm{PE}^a \longleftarrow \mathrm{RAM}}(j,l) = \mathrm{RAM}_{0_l}\left(\sum_{q=j+1}^{n-1}\frac{2^q}{2P} + t, \mathrm{PE}_p\right), \tag{4.3}$$

$$\mathrm{MAP}_{\mathrm{LLR}}^{\mathrm{PE}^b \longleftarrow \mathrm{RAM}}(j,pe_p,l) = \begin{cases} \mathrm{RAM}_{0_l}\left(\sum_{q=j+1}^{n-1}\frac{2^q}{2P} + t, \frac{2^j}{2} + \mathrm{PE}_p\right), & N_\lambda \leq P. \\ \mathrm{RAM}_{1_l}\left(\sum_{q=j+1}^{n-1}\frac{2^q}{2P} + t, \mathrm{PE}_p\right), & \text{otherwise,} \end{cases} \tag{4.4}$$

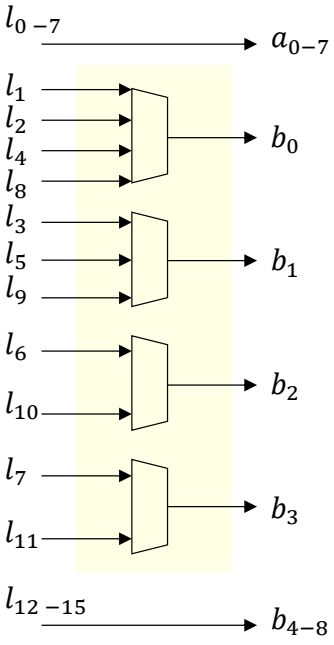
where a and b are the first and second LLRs input values to a single PE respectively.

Bypass buffer

At low decoding stages, the produced LLRs need to be immediately reused by the processing elements since they require only one clock cycle to be fully updated. Therefore, $L \times P$ buffer is used in order to process again these generated LLRs directly during the following clock cycle without causing a combinatorial loop in the circuit or adding extra latency to the decoding process. Thus, a set of MUX are used to select the appropriate LLRs to process from internal LLR RAMs or the bypass buffer.

j	≥ 3	2	1	0
$a_{0\to7}$	$L_{0\to7}$	$L_{0\to7}$	$L_{0\to7}$	$L_{0\to7}$
b_0	L_8	L_4	L_2	L_1
b_1	L_9	L_5	L_3	L_9
b_2	L_{10}	L_6	L_{10}	L_{10}
b_3	L_{11}	L_7	L_{11}	L_{11}
$b_{4\to7}$	$L_{12\to15}$	$L_{12\to15}$	$L_{12\to15}$	$L_{12\to15}$

(a)



(b)

Figure 4.3 – Intra-list selection network between LLRs of internal memories and PEs: (a) LLRs and PEs connection, (b) Selection network π-IntraL.

Pointer memory

The SCL decoder can also be seen as L SC decoder cores working in parallel with their own LLRs and memory resources. Nonetheless, the cores may share the same LLRs at some stages. Indeed, after each path selection, some of the initial L paths may be duplicated to give birth each to two new ones that differ from one another only in the last bit estimates. Moreover, they shall inherit all their LLRs related to the higher stages from the original path that gave them birth. If not discarded later, these paths keep sharing the LLRs relative to the stages that are not yet activated, when this is once done, each of them will produce its own LLRs. Thereby, to avoid copying the corresponding LLR memory, a pointer memory is introduced instead. Hence, each SC decoder core can read its inputs from one of the L RAM memories. To do so, a permutation network using a set of $2PL$ L-to-1 MUX called π-*inter-list* is implemented between internal LLR memories and PEs in order to select the appropriate LLRs. The introduced pointer memory is designed as a $(\log_2 N - 1) \times L$ register array. A register at row j and column l stores a pointer of $\log_2 L$ bits indicating the index of the RAM where the LLRs of path l at decoding stage j are stored. Initially, column l contents are all initialized with $l-$value stating that paths have

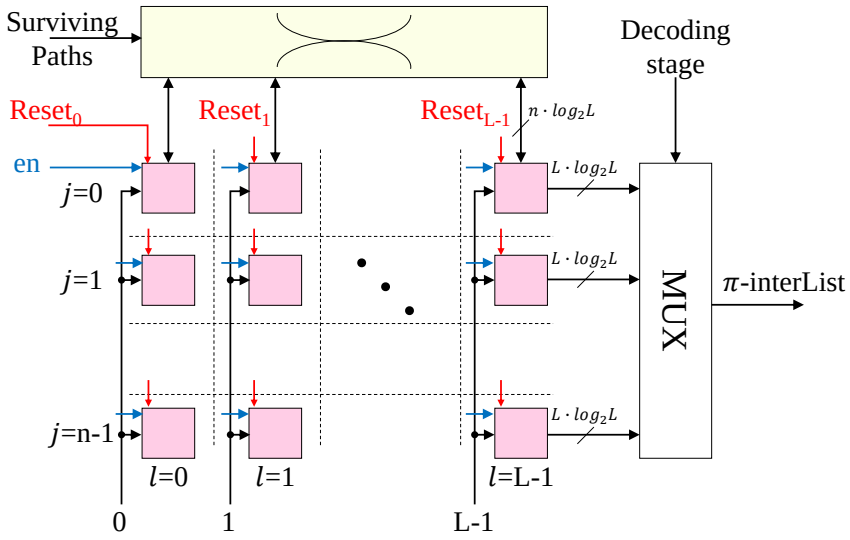


Figure 4.4 – Pointer memory architecture.

all their LLRs located in their respective own RAMs. During the decoding process, the pointers are updated in two different situations: (1) after path selection where the contents of the column l pointer memory are duplicated to column l' by means of $L \times L$ crossbar, and (2) after the decoding stage j is activated where each single path computes its own LLRs, the pointers of row j are reset to their initial values, i.e., implying that LLRs of path l at decoding stage j are stored in RAM_l. The structure of the pointer memory is illustrated in Fig. 4.4 where en and $Reset_l$ are the control signals that ensure the pointers are updated following the first and second situations respectively.

Path memory

Once the path metrics have been sorted and the surviving path identified, an array register of LN Flip-Flop (FF) is used to store the values of the L codeword bits. However, to avoid a high complexity search of information bits among the final codeword bits, only information and CRC bits are stored. The architecture of the path memory access is portrayed in Fig. 4.5. Being mutually independent, all FFs can be accessed simultaneously. This is made possible thanks to a N-bit enable vector. When a path l needs to be duplicated, all its so far decoded bits are copied to the registers that has been freed as their corresponding path has been discarded. Enable signals for the FFs are generated by the $\log_2 N$-to-N decoder which takes the currently decoded information bit index as input. To simplify the control, all the registers that are not accessed yet are enabled when

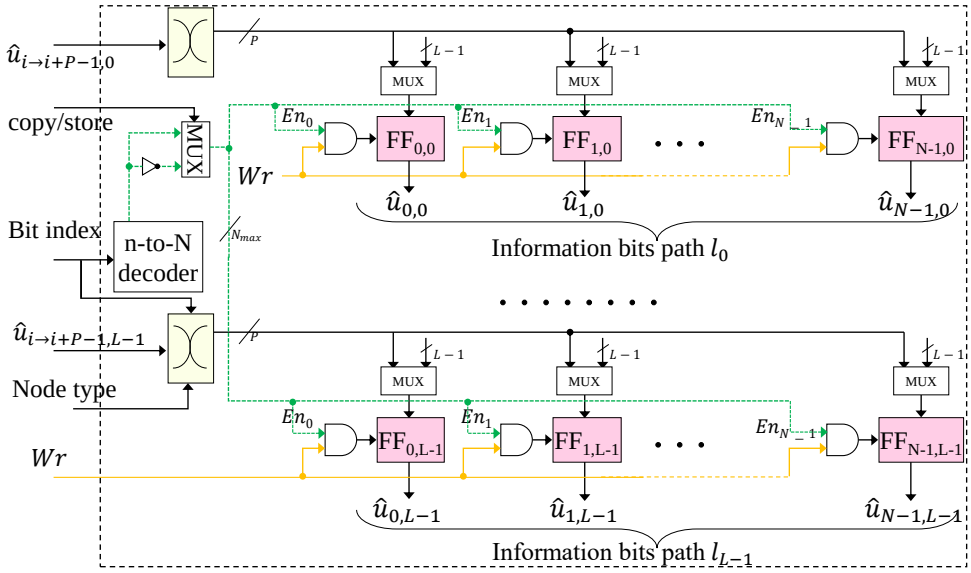


Figure 4.5 – Path memory access architecture.

storing input decoded information bits. This allows decoupling the registers that contain the previous decoded bits from the remaining ones. In this way, the same enable signals, when inverted, can ensure the copying of the surviving paths that occurs before storing the new bits. To do so, N MUXes are used to select between the copy/store operation to perform. Since by using a multi-bit decoding algorithm up to P information bits are stored at once, the FFs are virtually organized in N/P groups of P FFs, so that the bit of index p where $0 \leq p \leq P-1$ is connected to to the FFs of indices ϕ such that $\phi = p, P+p, \ldots, \left(\frac{N}{P}-1\right)P+p$. However, when reaching a tree node of length P that is not identified as one of the special node types considered by our decoder or any node other than R1 of size greater or equal than P, the number of information bits decoded in parallel has to be smaller than P. To manage this flexibility in decoding variable special node lengths, bits of a given path are routed to their respective FFs by means of a $P \times P$ crossbar.

4.1.2 Special nodes decoding

Instead of decoding every constituent code type individually, a special nodes list decoder (SNLD) which federates the common operations performed by the different special nodes is designed. Considering that the tree should be traversed sequentially with a node by node processing schedule, special node decoders are able to share most of the memory and

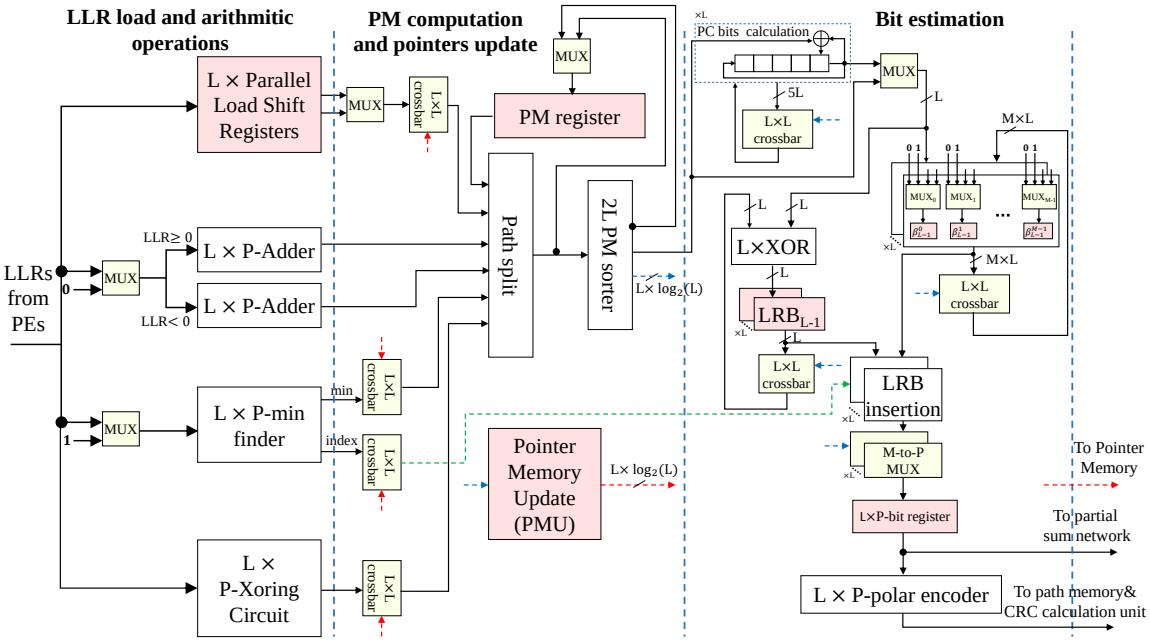


Figure 4.6 – Proposed SNLD architecture supporting different special node types.

the computational resources. Hence, hardware duplication is avoided. The architecture of the SNLD is portrayed in Fig. 4.6.

Since the R0 nodes comprise only frozen bits, no path splitting is needed to estimate the N_{R0} bits. It consists of summing up N_{R0} values. A tree-structure fully parallel adder is preferred to this purpose in order to guarantee no extra decoding latency. The complexity and the critical path of this adder structure are high especially when M is large. As long as $P \leq M$, the computations on the special node LLRs takes multiple clock cycles and a maximum of P LLRs can be processed at a time. Therefore, smaller tree-structure adder can provide the same computational speed as the fully parallel adder. To this purpose, a sum-accumulate operation is used instead, by introducing an accumulator register at the output of the adder in order to limit the depth of the tree from $\log_2 M$ to $\log_2 P + 1$ stages and to reduce the number of adders from $M - 1$ to P. As a result, N_{R0} values can be added up during $\lceil N_{\mathrm{R0}}/P \rceil$ clock cycles, where $\lceil x \rceil$ represents the closest integer value larger than x.

A Repetition node of length N_{Rep} is identified by the presence of one information bit and $N_{\mathrm{Rep}} - 1$ frozen bits. However a minimum of two steps are required for the estimate of the single information bit. The first one consists of updating the path metrics as part of path fork process including both 0 and 1 decisions on the single information bit of

the Rep node while the second consists of sorting the split path metrics to perform path selection. Unlike N_{R0} codes, two independent summations are needed by Rep node. Indeed, according to whether the bit estimate is considered to be 0 or 1, Rep decoder has to add up all the negative valued or positive valued LLRs, respectively. For the purpose of performing both summations in parallel, the R0 node adders are duplicated.

R1 nodes comprise only information bits which are decoded one-by-one as in the SCL algorithm. The speed-up lies in the fact that a set of bits is estimated at the root of the node. For each bit estimate, paths are duplicated, sorted and some of them are discarded. Since more than one bit is decoded in a row without going back to PEs, the SNLD comprises a *Pointer Memory Update* (PMU) unit that keeps track of the surviving paths from the beginning of each R1 and SPC nodes decoding.

A SPC node is identified by the presence of one frozen bit while all the remaining bits are information. These latter are estimated one-by-one as with R1 nodes. The least reliable bit is found as a first step for decoding a SPC node. It corresponds to the minimum LLR value at the top of the SPC node tree. An accumulation-minimum-finder tree-based structure similar to the one used for the adder units of R0 and Rep nodes with P comparators is used in this regard. The parity-check evaluation, in turn, is performed through an accumulator-XOR tree-based structure using P XOR gates. while the evolving even-parity constraint is performed for each of the existing L paths after each bit estimate through L XOR gates and stored in dedicated LRB registers. The final value of the latter is retained to preserve the even-parity constraint and inserted within the estimated bit vectors directly in its appropriate location provided by the minimum value index found in the first step (green arrow in Fig. 4.6).

Except for a classical bit decoding case, more than one LLR are admitted by the SNLD. Since they are immediately used, it would be costly in both latency and hardware complexity to write them in RAMs then read them back one-by-one as it is specified by the algorithm. However, they are stored in a bank of parallel-load shift registers, as illustrated in Fig. 4.7. Assuming $P \leq M$, they are seen as an array of M Q_i-bits registers grouped in M/P independent columns which can be accessed independently thanks to M/P enable signals. Q_i is the quantization level of the internal LLRs. LLR nodes are stored in different columns in the same order they are produced, starting from the rightmost column to the leftmost one during $\left\lceil \frac{2^j}{P} \right\rceil$ clock cycles. During the PM calculation phase, registers are shifted both horizontally and vertically until all the LLRs are used and bit-nodes are estimated during the decoding phase of R1 and SPC nodes as follows:

— Vertical shift: The LLR values implied in the estimation of R1 and SPC bits are used one-by-one by the decoder. They are accessed by shifting the P registers of the rightmost column from top to bottom whose outputs are connected to the inputs of the adjacent register.

— Horizontal shift: After $P-1$ vertical shifts, the next P LLRs to use in decoding, when they exist, are obtained by shifting registers of the different columns horizontally from left to right, where each register output is connected to the adjacent register input.

Excluding the last loaded register columns, the remaining ones can accept their LLRs from either their adjacent one or from input. This is made possible thanks to $P \cdot \left(\frac{M}{P} - 1\right)$ 2-to-1 MUX.

Since the least reliable bit in a SPC node is decoded first, its LLR value should be skipped whenever it is encountered. To do this, the LLR bank register in Fig. 4.7 is designed to output two adjacent LLR values instead of one. A MUX is used to select the output based on the least reliable bit index. When the PC bits are used in uplink, they are decoded using the length-5 cyclic shift register (Fig. 4.6). To support path competition, the generated intermediate paths read their values, i.e., LLRs, even-parity check and minimums by means of $L \times L$ crossbars. Newly estimated bits are temporarily stored in dedicated registers. A straightforward copy operation allows, by means of one $L \times L$ crossbar with port width M bits, to move all the contents of each of the L surviving paths from a register to another after each path selection. The process of PM sorting and bit estimation is repeated N_{R1} or $N_{\text{SPC}} - 1$ times. The PM register is updated directly from the $2L$ sorter in the same clock cycle. When all bits are estimated at their top tree, the source word bits are obtained through a polar encoder, the PMU updates the pointer memory and the search procedure of the next nodes in the polar code tree resumes. The partial sums needed to update LLRs through the g-function are computed and the CRC check process is initiated. In order to avoid extra latency, partial sums computations and g-function are performed in parallel.

4.1.3 Partial sum network

The decoded bits go through a XOR network in order to generate the appropriate partial sums required for the computation of the g function ((2.9)). Due to data dependency, the hard bit decisions must be generated at the same time as LLRs are computed in PEs.

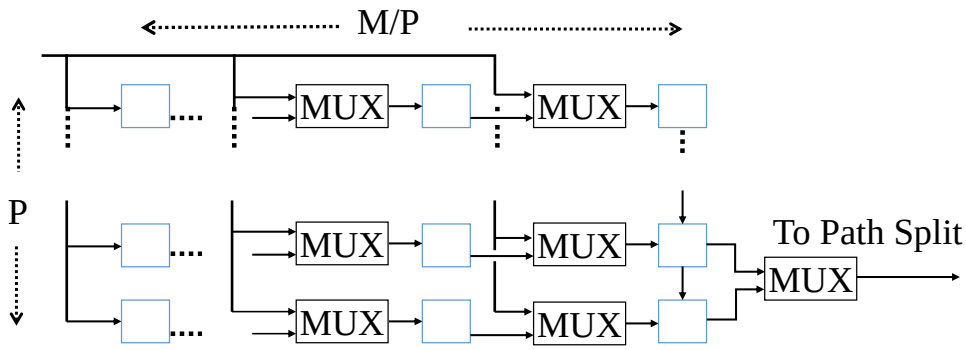


Figure 4.7 – Shift registers bank of the top-node LLRs.

Many types of PSN architectures have been devised for semi-parallel decoders [55], [100] [32] [63]. The List High-Performance (LHP)PSN proposed in [70] is composed of serial and parallel parts for area efficiency. Indeed, due to the semi-parallel decoder architecture, the parallel part allows only P partial sums to be updated in parallel instead of $N/2$ which are exclusively used at low stages g functions. However, at high stages, the partial sums are serially generated in the serial part from previously produced partial sums. To do so, L dual-port RAMs of $N/2$ bits each are used to store partial sums of the higher stages of each path. The newly updated partial sums are obtained using a XOR array of P gates and results are stored back in the RAMs to be used in future partial sums generation. Since candidate competition in SCL algorithm causes frequent discarding and duplicating of codeword paths, instead of copying PS of a duplicated paths from one RAM to another corresponding to a discarded path, a specific pointer memory register array similar to that used for LLRs is used (see Fig. 4.4). In this way, for each decoding stage, PS of a given path can be found in any of the L RAMs. PSN is provided with a crossbar that contains L L-to-1 multiplexers to supply the L sets of PEs with the appropriate PS. The same architecture of the PSN serial part described in [70] is used here. In the parallel part of the LHPPSN [70], the PS are computed only for the g function that occurs at low stages ($j \leq \log_2 P$). To this purpose, for each path l, P registers are used to update the partial sums generated from special nodes of maximum length of P. When the length of the decoded special node is higher than P, the partial sums are directly generated from the SNLD and do not need to be computed in the parallel part. They are directly fed to the PEs. In order to accommodate candidate competition that occurs between successive partial sum updates of nodes smaller than P, $L \times L$ crossbar with port width P bits is implemented to copy the surviving paths to the registers that have been freed in the meantime. When the decoding stage j such that ($j > \log_2 P$) is activated, the content of

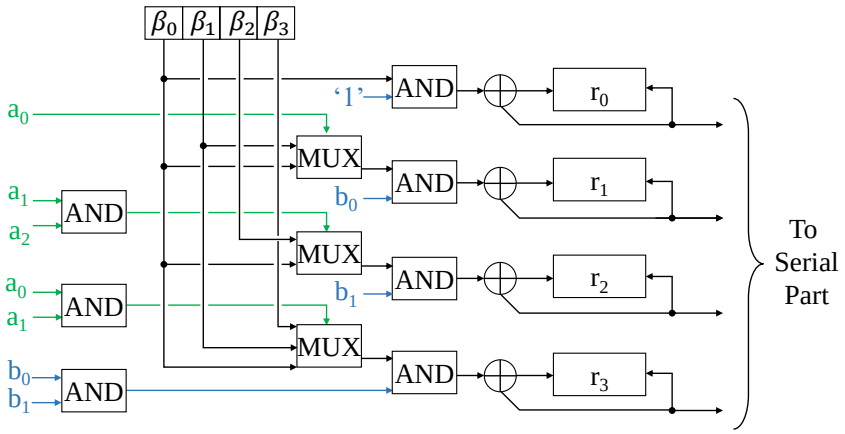


Figure 4.8 – Parallel part of LHPPSN supporting multi-bit decoding for $P = 4$.

the P registers of the parallel part are used in the PE and stored in RAM of the serial part to be next used in the generation of other high stages partial sums. These registers are then reset. In [70], the source input to the parallel part of the PSN is supposed to be 1-bit length, an array of ANDs and XORs logic gates is used to generate the partial sums after each input. The set of partial sums that need to be updated following the decoding of the i-th bit is obtained thanks to a G row vector derived from G_N as described in [32]. To support multi-bit decoding in which the source input is a vector of P bits, the parallel part of the LHPPSN has been adjusted. The proposed architecture is depicted in Fig. 4.8 for $P = 4$. Since the special nodes considered in our decoder are length-variable, the actual number of input bits following the decoding of a special node is also variable and is equal to 1, 2 or 4 following the example where $P = 4$. Therefore, if the input bits are smaller than P, only part of the P PS needs to be updated and the registers that store these PS need to be identified. To do so, two parameters are to be taken into consideration. First, the size of the special node (a_2, a_1, a_0) which will determine the correct bit-location within the input vector β. Second, the first $\log_2 P$ bits (b_1, b_0) of the index i, $(0 \leq i \leq N-1)$ that correspond to the index of the first decoded bit in the current special node.

4.1.4 CRC calculation

The CRC bits are scrambled and interleaved in downlink prior to encoding as explained in 2.4. The interleaving of CRC bits does not foster a straightforward polynomial CRC check implementation. However, the same interleaved CRC generator matrix used to provide and interleave CRC bits, the CRC bits locator and scrambler provided by the

polar code construction are in turn used to check CRC bits. The architecture of the CRC check operation is presented in Fig. 4.9.

A CRC checksum vector defined as the modulo-2 operation of the CRC generator matrix and an all-one L_{CRC} vector is initialized before the decoding process starts which is particular to the polar code information length A. It is then updated each time an information or a CRC bit is encountered. K XOR gates are needed to update it in one clock cycle. The CRC checksum vector is specific to each path. Therefore, it is subject to duplication and discarding processes. a CRC checksum bit is compared to its corresponding index in scrambling pattern vector when the last 1-valued bit in the i-th line of the interleaved CRC generator matrix is reached by means of AND gate. If the comparison disagrees then the CRC check has failed. A combinational logic block is introduced to check whether the currently decoded bit is indeed a CRC bit or not. When using the muti-bit decoding algorithm, up to M bits can be produced each time the SNLD is activated while a subset with P bits can be produced simultaneously. The lowest amount of clock cycles between two consecutive activation of SNLD is equal to four and occurs when two consecutive special nodes share the same parent, i.e., they are the left and right direct children of the same parent. Thus, in order to avoid extra latency, all the produced bits by the SNLD shall go through the CRC verification in at most $\frac{M}{P} = 4$ clock cycles. This is always true if all the P decoded bits are handled in parallel during the CRC check operation. Therefore, a P-parallel checksum update is implemented to this end. Moreover the CRC checksum of a set of P bits depends upon the last 1-valued information bit which in turn depends on the previous one and so on. The toggling property of the XOR operation is the reason why a CRC check fail may occur but in fact is hidden due to successive clearing, justifying the need to keep track of all the intermediate checksums. A P-recursive structure of $K \times P$ XOR gates is implemented to this end. As a result of this dependency, the recursive structure leads to a long critical path and hence a low throughput. The validity of path candidate is indicated by the CRC state register that keeps track of whether any of the checks associated with the CRC bits has failed. it is updated by the $P-$ AND array. Depending from the special node type and size, information and CRC bits may be located anywhere between the M decoded bits. The currently decoded information bit indices are provided and additional combinational logic is however added to manage finding the location of information bits within the P decoded sequence. Note that none of the codeword candidates shall be pruned from the list of the L surviving paths, even though we know that they will fail the CRC or have already

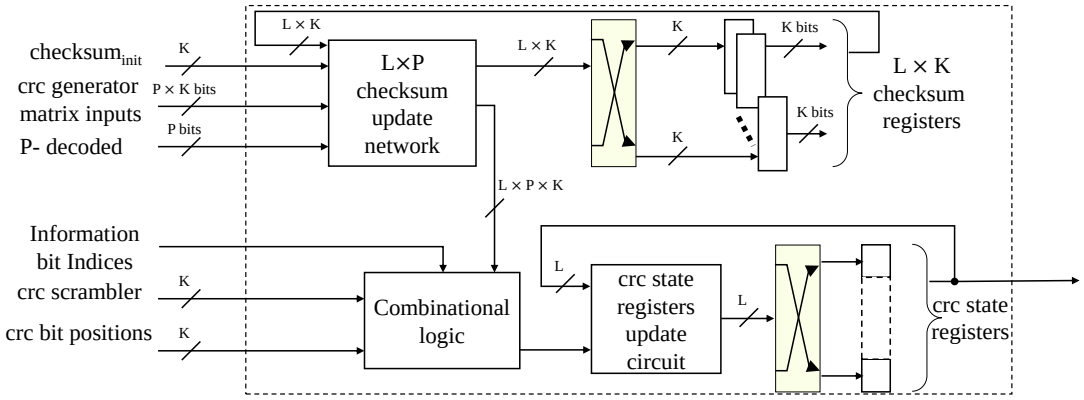


Figure 4.9 – Architecture of the proposed parallel CRC check.

failed it with respect to the CRC state registers. The decoding of all the L competing candidate paths should continue, otherwise the error detection capability brought by the CRC will be damaged.

4.1.5 Proposed on-the-fly rate-flexible decoding of polar codes

To apply dedicated special node decoders for the 5G NR polar code, the types and sizes of the nodes within the current frame need to be provided. The large flexibility in frame and code rate leads to a large number of different frozen bit sets. This increases the number of different special nodes, making it difficult to store in memory their number and positions. To tackle this issue, we propose a new method to identify the different special node structures (R0, Rep, SPC and R1) on-the-fly directly in hardware without the need to store any list in memory. Their determination is done from the frozen set and by merging different lower size special nodes into a new special node type of larger size. Whereas, the node identification circuit proposed in [49] uses the bit-channel relative reliability vector and should be instantiated at every decoding stage of the Layered-Partitioned SCL [49].

In order to illustrate our approach, let a_0 and a_1 be frozen set bits and v_0 and v_1 be the two bits of the vector indicating the special node type. The sequence of v_0 and v_1 is 00 , 01 , 10 and 11 referring to R0, Rep, SPC, and R1 nodes respectively. For hardware optimization purposes, we define a new special node type corresponding to the only case where two bits from the frozen set do not represent any of the particular constituent code listed above, i.e., the sequence $\{$*information , frozen*$\}$ and is called *No type* node. Henceforth, the sequence 10 ($v_0 = 1$ and $v_1 = 0$) refers to both *No type* and SPC nodes. However, assuming having additional information on their size, these two special nodes

cannot be overlapped. Indeed, SPC nodes start to be considered from length-4 patterns by merging two special nodes of length greater or equal to two. Table 4.1 summarizes the logic equations that identify the different above-mentioned special nodes.

Table 4.1 – Identification of the different special nodes based on the frozen set couple a_0a_1 for special nodes of length two, and based the on the 2-bit vectors v_0v_1 and $v'_0v'_1$ indicating the type of the two nodes to merge for special nodes of length higher than two.

Node type	R0	Rep	R1	*No type*/ SPC
$M = 2$	$\bar{a_0}\bar{a_1}$	$\bar{a_0}a_1$	a_0a_1	$a_0\bar{a_1}$
$M = 4$	$\bar{v_0}\bar{v_1}\bar{v'_0}\bar{v'_1}$	$\bar{v_0}\bar{v_1}\bar{v'_0}v'_1$	$\bar{v_0}v_1v'_0v'_1$	$\bar{v_0}v_1v'_0v'_1$
$M \geq 8$				$v_0\bar{v_1}v'_0v'_1$

Starting from a given frozen set, the structure of Fig. 4.10a is capable of determining the different special node types of length two. For higher length nodes, the structure of Fig. 4.10b intends to merge different lower size special nodes into a new special node type of larger size based on the 2-bit vector v_0v_1 and $v'_0v'_1$ indicating the type of the two nodes to merge. With one difference, a SPC node of length four is obtained by merging R1 and Rep nodes of length two while a SPC node of length greater than four is obtained by merging a R1 node and another SPC node (highlighted in red), hence the presence of the MUX. Therefore, the two structures of Fig. 4.10a and Fig. 4.10b represent the building blocks of the identifier intended to determine the different supported special node types of any length.

To determine all the special node types for a polar code of length N and special nodes of maximum length M, $N/2$ identifications of special nodes of length two (Fig. 4.10a) followed by $\sum_{m=1}^{\log M-1} 2^{\log M-(m+1)}$ identifications of special nodes of length greater that two (Fig. 4.10b) are performed. The PC(8,4) given as an example in Fig. 4.11 comprises a length-4 Rep node and a length-4 SPC node, highlighted in red. To obtain the list of all existing special node types and sizes for this code, four instances of the structure depicted in Fig. 4.10a and three instances of the structure depicted in Fig. 4.10b are needed for a full parallel search. The output v_0v_1 (in blue) of each identification module, which corresponds to the type of the special node, is appended by three additional bits to form the vector $V = [v_0, v_1, v_2, v_3, v_4]$. The two bits v_2, v_3 (in black) indicate the size of the encountered node while v_4 (in green) is a flag that tells whether it is a valid special node or not.

The set of instantiated modules used for determining the special nodes of length 2

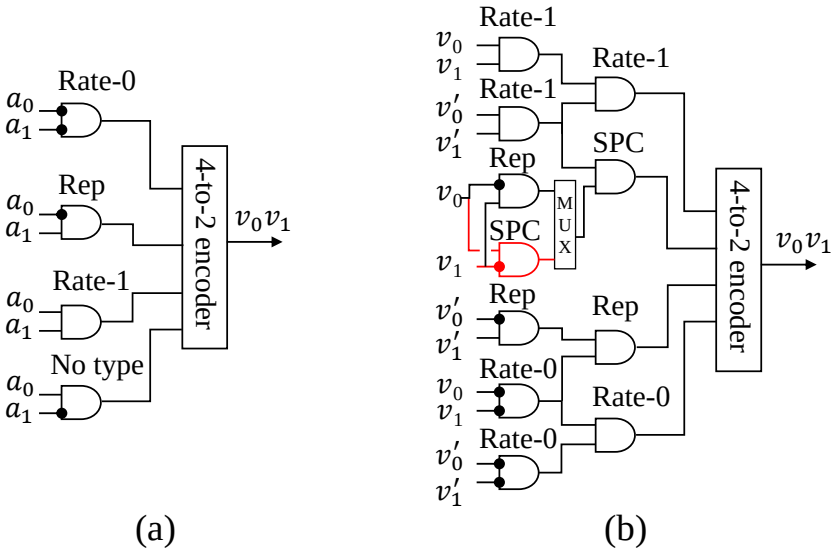


Figure 4.10 – Proposed module for identification of constituent codes of different length: (a) identification of the different special nodes of length two, (b) identification of the different special nodes of length four and above.

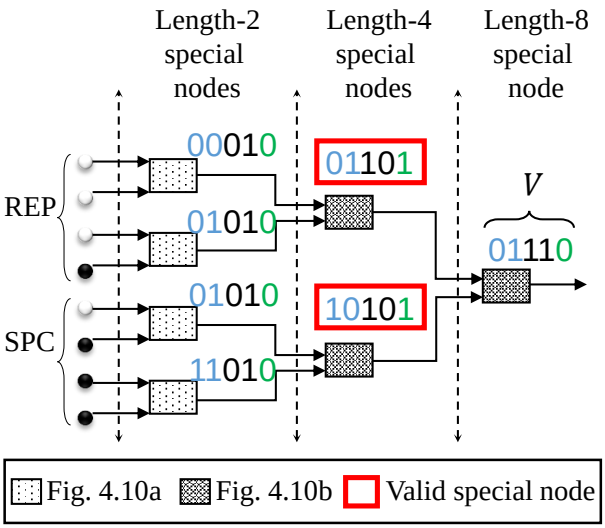


Figure 4.11 – Identification of special nodes illustrated for a PC(8,4). This corresponds to the architecture of an M-SNI with $M = 8$.

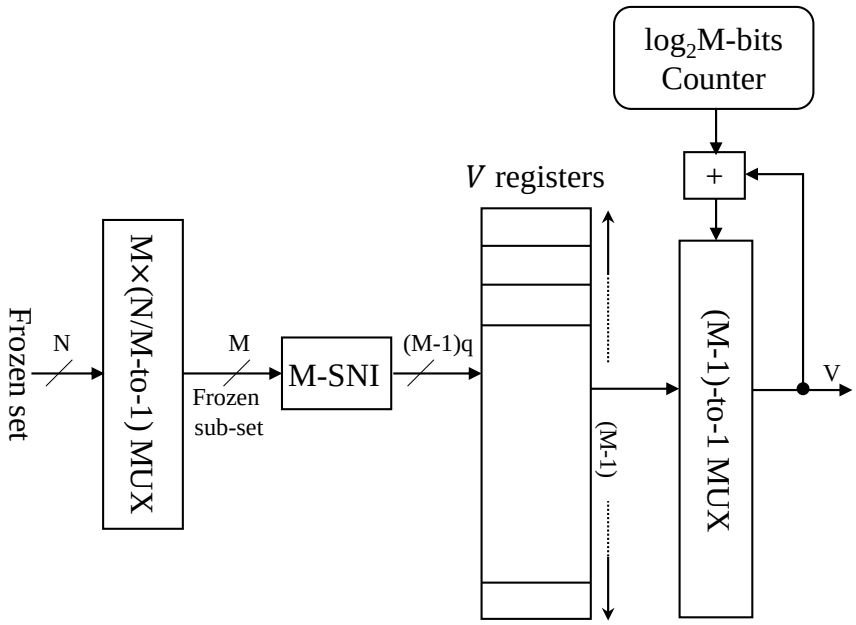


Figure 4.12 – Architecture of the proposed on-the-fly identification of special nodes (SNI).

to length M (Fig. 4.11) is called M-Special Node Identifier (M-SNI). Therefore, for a polar code of length N and special nodes of maximum length M, $\frac{N}{M}$ M-SNI are required for identifying in a single step all the existing special nodes. The identified special and non-special nodes are stored together in register arrays in the same order as they are searched during the decoding process. This reduces the complexity of multiplexing the special node vectors V compared to the case where only valid ones are retained. Since the successive cancellation decoder is sequential, the V vectors are read from the register one at a time through a $\frac{N}{M} \cdot (M-1)$-to-1 MUX. A $\log_2 N$-bit counter is used to generate the register-based memory addresses. At each special node iteration search, the next vector to read from the set of registers depends on the size and address of the previous one. The complexity of the proposed architecture increases linearly with the code length N. However, since the special nodes are searched only upon request, a low complexity serial implementation is favoured. Thanks to the sequential nature of the SC-based decoding algorithms, a single M-SNI used serially is able to produce the same results as the full parallel search of special nodes with less hardware resources. Therefore, $M \times (\frac{N}{M})$-to-1 multiplexers are used to process serially the frozen set bits in groups of M. The proposed architecture for on-the-fly serial search of special nodes is depicted in Fig. 4.12. We refer to this proposed architecture as Special Node Identifier (SNI).

4.1.6 Control unit

The control signals needed by the different components of the decoder architecture are provided by the control unit. It includes a Finite State Machine (FSM) that defines the various phases of the decoding process. Several counters are used to generate the addresses of the required memories previously described. To simplify the control, the PSN memory addresses are also stored in a dedicated memory. The control unit includes a couple of adders to update the information bit index after decoding a special node or an information bit and to increment the bit index i. It also includes a shift register to determine the current decoding stage. In order to manage the different control signals of all the components supposed to operate in parallel to the main decoding process and may last longer than one clock cycle depending on the length of decoded bits in SNLD, pipeline registers are used. They first allow delaying CRC check, path memory writing and PSN control signals and then hold these signals active as long as necessary during the different states of the FSM. Finally, the control unit keeps checking if at least one codeword candidate validates the CRC check, otherwise it triggers the early termination of decoding.

4.2 Results and performance analysis

The devised decoder architecture has been described in VHDL using generic forms for flexibility and scalability purposes. Application-specific VHDL packages that include key functions and generic quantization signal types have been defined to support this genericity. The validation of the decoder was done for each code length and for both uplink and downlink polar codes through VHDL testbenchs. To do so, a Matlab software implementation compatible with the design specifications supported by the proposed decoder was developed and used as a reference during the validation process. It generates the channel LLRs for the hardware testbench and performs as well the fixed point decoding of the polar codes in software. At the end of the decoding process, the output of both hardware and software decoders are compared.

4.2.1 Synthesis results

The proposed hardware architecture for decoding 5G NR polar codes has been implemented on a Xilinx Virtex 7-xc7vx485t FPGA. Based on the quantization study presented

Table 4.2 – FPGA synthesis results of the proposed 5G NR polar decoder

Block name	LUTs	FFs	BRAMs
Soft data part (SDP)	8414	4672	64
Hard data part (HDP)	34533	11652	4
Special nodes list decoder (SNLD)	7482	3839	0
Special node identifier (SNI)	442	78	0
Control unit (CU)	469	223	0

in Section 3.1.3, the number of bits used to represent internal LLR, PM and channel LLR values is $Q_i = 6$, $Q_p = 7$ and $Q_c = 4$, respectively. Eight PEs are instantiated in the architecture for each of the L paths, where $L = 8$. The maximum size of special nodes M is set to 32. Internal LLRs and partial sum bits are stored in the FPGA dual-port Block RAMs (BRAMs) while input LLRs and partial sum memory addresses are stored using look-up tables (LUTs).

To analyze the impact of each hardware unit on the final hardware complexity, the two parts SDP and HDP of Fig. 4.1 besides the special nodes list decoder, special node identifier and control unit are synthesized separately and presented in Table 4.2. The results show that the HDP occupies 67% and 57% of used LUTs and Flip-Flops (FF), respectively. The high number of resources needed in this part comes from the path memory unit where $N \times L$ 2-to-1 MUXes are used to perform copy/write operations of the decoded bits stored in $N \times L$ FFs to allow the competition between codewords to populate the list all along the decoding process. Furthermore, the SCL decoder relies on interconnect networks to apply path selection function and routing of internal LLRs for each of the L SC decoders. This significantly impacts the overall complexity. The total number of BRAMs used to store the internal LLRs is 64, configured as 128 × 18kb RAMs, which corresponds to the 8 pairs of RAM depicted in Fig. 4.2. On the other hand, the RAMs used to store the partial sums require 4 BRAMs configured as 8 × 18Kb RAMs.

Decoding latency is the main critical performance metric when considering the 5G NR polar codes that protect the control channel. A target end-to-end latency of 0.5 ms implies a physical layer latency of 50 μs [34, 67]. Flexibility and low latency are the driving priority for the design of this channel decoder, which is the main demanding component of the physical layer. For 5G NR polar codes, the latency for decoding one codeword is not constant and is highly affected by the choice of the operating code length and code rate. In order to give a full analysis of the latency, all combinations of code lengths and

Table 4.3 – Average and maximum latency measured for the proposed decoder.

			Worst-case latency		Average Latency	
	N	# code	cc	[μs]	cc	[μs]
Downlink	64	435	146	1.35	121	1.11
	128	3451	284	2.63	207	1.91
	256	9452	483	4.47	376	3.47
	512	2286	780	**7.22**	719	6.65
Uplink	1024	3491	2583	**23.91**	1914	17.72

code rates ranging from $R = 1/8$ to $R = 5/6$ are evaluated through simulations. Table 4.3 provides the average and the worst-case latency for decoding one codeword. The latency is measured in number of clock cycles (cc) and in μs considering the maximum clock frequency $f_{\max} = 108$ MHz. The results show that the maximum latency recorded by the decoder is 23.91 μs which is 2.1 times lower than the physical layer latency constraint. This worst-case latency appears in the uplink scenario, whereas it is only 7.22 μs in the downlink.

4.2.2 Comparison with state-of-the-art FPGA implementations

Logic synthesis and performance results are summarized in Table 4.4. Implementations that are fully compatible with 5G code specifications, in terms of code structure and flexibility, allow for direct and fair comparisons. Therefore, the recently available Xilinx polar decoder [95] is the most relevant reference with respect to the available decoder designs in the literature. Performance results of Xilinx decoder are available, yet with no published details on the architecture. Compared to this decoder, our proposed architecture has 60% and 70% less decoding latency for the uplink (1024,512) and the downlink (512,40) polar codes, respectively. Our design consumes 38% less Flip-Flops (FFs), but 11% more LUTs and 25% more BRAMs. The throughput, however, does not compare favorably, yet still compliant with the 5G NR requirement for this code used for the control channel.

In order to extend the comparison, we have considered recent designs that targeted FPGA implementation with similar code length, yet not compliant with 5G NR polar codes. For that, a second configuration of our proposed architecture has been designed and synthesized with $L = 4$ while keeping the number of PEs per list unchanged. Compared to the folding polar decoder of [61], our decoder uses 74% less LUTs and 6% less FFs

Table 4.4 – Comparison with existing FPGA-based SCL Architectures.

Decoder	[61]	[96]	[60]	**This work**	Xilinx [95]		**This work**	
List size	4				8			
Algorithm	SCL	SCL	SSCL[1]	**SSCL[2]**	NA		**SSCL[2]**	
Flexibility	Limited	Limited	Limited	**High**	High		**High**	
Quantization	NA	5	NA	**(4,6,7)**	8		**(4,6,7)**	
# PE	64	64	64	8	NA		8	
f_{op} (MHz)	N/A	42.66	445.2	**108**	223		**108**	
FPGA Device	stratix V	Kintex 7	stratix V	**xc7vx485t**	xc7vx485t		**xc7vx485t**	
ALMs/LUTs	101160	142961	8146	**26049**	45569		**51262**	
FFs	13544	19795	2862	**12603**	33063		**20270**	
BRAMs	0	0	0	**34**	51.5		**68**	
(N,R)	(1024,0.5)	(1024,0.5)	(1024,0.5)	**(1024,0.5)**	(512,0.07)	(1024,0.5)	**(512,0.07)**	**(1024,0.5)**
Latency (μs)	4064[3]	8.9	1177	**10.73**	11.35	46.41	**4.76**	**18.37**
TP (Mbps)	N/A	40.93	0.452	**47.7**	28.68	88.4	**8.4**	**34.4**

[1] Stochastic SCL with 2-level decoding
[2] Simplified SCL with four constituent codes
[3] Number of cycle cycles

while decoding the (1024,512) code in less than half the time (converted in clock cycles). However, without any reported clock frequency, the latency comparison is not complete. Compared to the stochastic SCL decoder of [60] targeting wearable and IoT devices with strict hardware constraints, our decoder supporting rate and frame size flexibility exhibits 109 times less latency while requiring 3.2 times and 4.4 times the numbers of LUTs and FFs, respectively.

4.3 Multi-frame decoding

In the previous chapter, we have shown the impact of P on the decoding latency of polar codes in particular those selected in 5G NR. We have come to the conclusion that the utilization rate of PEs is relatively low in particular at low decoding stages where the ratio of used PEs to P tends to zero especially when P is high. In this section, we aim to improve the hardware utilization of the proposed decoder using the first multi-frame decoding approach presented in Chapter 3 based on the re-use of the available hardware resources to increase the throughput of the decoder. To do this, we take advantage of the flexibility required by the 5G NR polar codes in terms of block length to use this feature of the decoder to support the parallel decoding of two frames of lengths $N \leq 512$. The proposed decoder architecture is similar to the one proposed earlier in this chapter with additional improvements, in particular in terms of memory storage management. Therefore, it preserves all the features of flexibility required in the context of 5G NR polar

codes. Thus, the same memory structure, computational and logical operation units are used for the different storage needs and for LLR update rules, partial sums and CRC check computation. However, some of these functional components and memory resources are duplicated to support concurrent decoding of two polar code frames without increasing the control complexity of the decoding.

Following this approach, the memories used to store the different types of data and metrics of the polar decoder are either duplicated or managed to support the multi-frame decoding process. In this design, we propose to duplicate partial-sum memory as part of duplicating the entire partial-sum network described in Section 4.1.3 and to duplicate the pointers memory to keep track of the surviving L codewords for each decoded frame. Furthermore, the same memories designed in Section 4.1.1 are used to store LLRs of both frames. The depth of the internal memory required to store these LLRs is:

$$MEM_{LLR} = T \cdot \sum_{j=p+1}^{\log_2 \frac{N}{T}-1} 2^j/2P + T \cdot \sum_{1}^{p} 1 = \frac{N_{max}}{2P} + T \cdot (\log_2 P - 1)\,, \tag{4.5}$$

where, T, the number of frames decoded in parallel, is equal to two.

The number of FFs dedicated to store the estimated bits in the former decoder is sufficient to store the estimated bits of the different frames. Therefore, a new control management is introduced for this purpose. A generic architecture of the path memory access is proposed, in which the N FFs used to store codeword bits of a polar code frame of length $N = 1024$ bits are used to store the codeword bits of multiple frames of length ($N/T \leq 1024$). The path memory access to a single entrance list for $T = 8$ is illustrated in Fig. 4.13. The set of FFs is divided in T groups of N/T. Each group is associated with a $\log_2(N/T)$-to-N/T decoder that provides it with enable signals. A combinational circuit is used to generate the write signal to select the group(s) of FFs that are addressed in a given time depending on the frame index and length.

The decoder architecture offering the capability of decoding two frames according to the proposed multi-frame decoding approach is designed. With this architecture, the decoder is still compliant with the 5G NR polar codes in terms of code length and code rate flexibility. This architecture is then described in VHDL. Synthesis results on Xilinx Virtex 7-xc7vx485t FPGA device are presented in Table 4.5 for one and two-frame decoding architectures. The number of bits used to represent internal LLR, PM and channel LLR values is $Q_i = 6$, $Q_p = 7$ and $Q_c = 4$, respectively. Eight PEs are instantiated in the architecture for each of the L paths, where $L = 8$. The maximum size of special nodes M

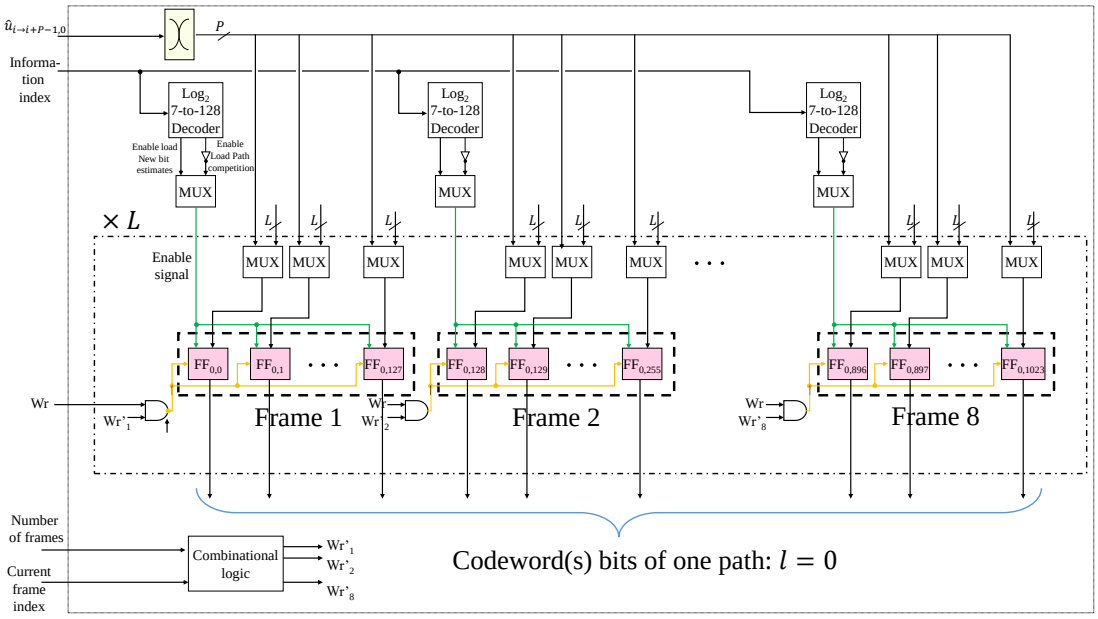


Figure 4.13 – Path memory access architecture of multiple polar code frames.

Table 4.5 – Synthesis results of the proposed decoder in FPGA for different value of T.

T	LUTs	FFs	BRAMs
1	50793	20047	68
2	52982	21748	72

is set to 32. The synthesis results do not include the control unit. Nonetheless, all the control signals required by the decoder are included into the architecture sub-block and are seen as external signals of the top-level decoder architecture by the synthesis tool.

As we dedicate a partial sum network unit for each frame including partial sum memories, we note that the number of 18kb BRAMs is increased by $L = 8$. It should be noted that 68 BRAMs would be enough to store partial sums of both frames if control management was introduced to manage partial sum memory addresses. When it comes to LUTs, we see that their number is increased by only 4.3% when compared to the original proposed decoder. This increase corresponds to the duplicated CRC unit and partial-sum network in addition to the added multiplexing resources required by the decoder. Finally, the number of required FFs is increased only by 8.4% when compared to the originally proposed decoder.

The latency for decoding one and two frames with the proposed decoder was measured

Table 4.6 – Latency and throughput of decoding one frame and two parallel frames with the proposed decoder for different code lengths and code rates.

N	R	$T = 1$		$T = 2$	
		Lat.(μs)	TP(Mbps)	Lat.(μs)	TP(Mbps)
128	1/4	1.68	20.50	2.38	28.93
	1/3	1.96	23.65	2.8	32.67
	1/2	2.08	33.17	3.13	44.04
	2/3	2.49	36.85	3.84	47.78
256	1/4	3.65	18.89	5.11	27.04
	1/3	3.55	25.81	5.49	33.43
	1/2	3.88	35.54	6.12	45.17
	2/3	4.68	38.95	7.52	48.49
512	1/4	6.28	21.98	9.60	28.79
	1/3	6.88	26.49	10.80	33.78
	1/2	7.75	35.63	12.27	45.03
	2/3	9.22	39.58	14.90	48.97

for various code rates, in particular R =1/4, 1/3, 1/2 and 2/3, for the 5G uplink polar codes of lengths N =128, 256 and 512. Latency results and corresponding information throughput considering the maximum clock frequency $f_{\max} = 108$ MHz are provided in Table 4.6. As expected, the latency of decoding one frame is increased when two frames are decoded in parallel. For the selected code rates, the increase in latency ranges from 41% to 54%, from 39% to 60% and from 52% to 61% for $N = 128$, 256 and 512 bits, respectively. On the other hand, the throughput is increased as also expected. This increase ranges from 29% to 41%, from 24% to 43% and from 23% to 30% for $N = 128$, 256 and 512, respectively.

4.4 Summary

In this chapter we proposed an original hardware architecture for decoding the 5G NR polar codes of the uplink and the downlink control channels. We have detailed all the different elements that compose the proposed architecture, in this case, the memory structure, the computational units and the permutation networks. An original special node list decoder and multi-bit calculation of partial sums and CRC were also proposed and detailed in this chapter. Thanks to a special node identifier, the proposed decoder continues

to benefit from tree pruning techniques to speed-up the decoding whilst maintaining compliance with the 5G NR and the various defined combinations of code rate and code length. The proposed decoder was described in VHDL, validated and synthesized for FPGA target technology. A thorough analysis of the key performance indicators of the decoder, including decoding latency, throughput and FPGA utilization resources was provided and comparisons with state-of-the-art polar code FPGA implementations were conducted. Indeed, measured throughput and latency values of the proposed decoder obtained with an FPGA target are able to meet 5G requirements. Moreover, synthesis results have shown a hardware efficiency that compared favourably with state-of-the-art FPGA implementations of polar codes. Our proposed decoder design reduces decoding latency and complexity compared to the recently available Xilinx polar decoder, the only published fully-compliant 5G NR. Finally, we have proposed a new way of decoding multiple frame of polar codes on the basis of the proposed decoder architecture with only a minor modification that still preserves the decoder flexibility. A straightforward implementation of this idea has been performed and results have been discussed.

Chapter 5

Conclusion and future work

Conclusion

The work presented in this manuscript targeted the investigation of the impact of main code and decoder design parameters on the latency, throughput, and the hardware complexity of semi-parallel decoding architectures and the proposition and implementation of an original hardware architecture for decoding the 5G NR polar codes.

In Chapter 2, the basic concepts of polar codes have been reviewed together with detailed presentation of the successive cancellation decoding algorithm and the different recent variants proposed in the literature to improve the decoding performance. Particular emphasis has been placed in this chapter on tree-pruning techniques, motivated by their positive impact on latency and throughput. The chapter has introduced, in addition, the specification of the polar codes adopted in 5G NR, necessary for comprehending the contributions proposed in the subsequent chapters.

In Chapter 3, the simulation model developed to integrate the full encoding and decoding scheme of the polar codes of 5G NR is presented alongside the communication channel and the different operations of interleaving and rate-matching. This software model was used to evaluate the performance of 5G polar codes under different types of fast decoding SC algorithms and was used to evaluate the performance of the decoder architecture with different parameters related to quantization and parallelism. The design space of the semi-parallel polar decoder was explored in terms of algorithm parallelism

choices and hardware complexity. Both algorithmic and architectural design levels were taken into consideration in addition to various tree-pruning techniques in order to evaluate the relationship between hardware complexity and key performance indicators of polar decoders. In particular, the latency and throughput performance of the SC-based decoders were targeted. Moreover, the activity of the different processing units of the SC-based decoders, including processing elements and special node decoders, was evaluated. The possibility of decoding multiple codewords with a minimum amount of added resources while preserving the flexibility of the architecture was studied and a suitable solution was provided. This latter was motivated by the resulting improvement in the decoding throughput and the need for devising an efficient solution for blind decoding particularly for the physical downlink control channel.

In Chapter 4, an original hardware architecture for decoding the 5G NR polar codes of the uplink and the downlink control channel was proposed. We have detailed all the different elements that compose the proposed architecture, i.e. the memory structure, the computational units and the permutation networks. An original special node list decoder, multi-bit calculation of partial sums and CRC check were also proposed and detailed in this chapter. Thanks to a special node identifier, the proposed decoder continues to benefit from tree pruning techniques to speed-up the decoding whilst maintaining compliance with the 5G NR and the various defined combinations of code rates and code lengths. The proposed decoder was described in VHDL, validated and synthesized for FPGA target technology. A thorough analysis of the key performance indicators of the decoder, including decoding latency, throughput and FPGA utilization resources was provided and comparisons with state-of-the-art polar code FPGA implementations were conducted. Indeed, measured throughput and latency values of the proposed decoder obtained with an FPGA target are able to meet 5G requirements. Moreover, synthesis results have shown a hardware efficiency that compared favourably with state-of-the-art FPGA implementations of polar codes. Our proposed decoder design reduces decoding latency and complexity compared to the recently available Xilinx polar decoder, the only published fully-compliant 5G NR. Finally, we have proposed a new way to decode multiple polar code frames simultaneously. While it is based on the proposed decoder architecture, it introduces only a minor modification that still preserves the flexibility of the decoder. A straightforward implementation of this idea has been performed and results have been discussed.

Future work

The research conducted in this work showed that it was challenging to satisfy simultaneously decoding latency, extreme frame size and code rate flexibility, decoding performance and efficient implementation constraints taking into account the short history of polar codes. However, motivated by their adoption in 5G, the scientific community has made and is still making important progress in this regard. This work constitutes a contribution to this effort. Indeed, the proposed architecture is capable of operating in μs orders with a relatively low decoding complexity and hence can fit the stringent constraints required by the standard that selects polar codes to cover the control channel. However, extensions and improvements are still possible in regards of the performed work. Below is a list of suggested research study items:

1. A way forward following the results of Chapter 3 is to study and analyse the impact of decoding other new special nodes [40] and generalized special nodes [22] on latency, complexity and throughput. The analysis conducted in that chapter targeted the set of 5G NR polar codes. Consequently, limited code and decoder design parameters were considered. Therefore, future work could take into consideration variable list sizes, larger polar codes and different sets of polar code constructions. Moreover, other decoding algorithms such as SCF and BP could be considered for a broader comparison and analysis.
2. In the last parts of Chapter 3, we have investigated the simultaneous decoding of multiple frames of polar codes through the proposition of two efficient multi-frame decoding approaches. A straightforward implementation for parallel decoding of two frames, that uses alternately the same PEs and special nodes, of the same polar code was designed. The next step, would be to design a new multi-frame decoding architecture implementing the Fast-SSCL algorithm. Such an architecture based on the second approach presented in Chapter 3 consists of duplicating the number of PEs while using a unique special node decoder would significantly increase the throughput while benefiting from the flexibility provided by the proposed decoder in Chapter 4. Furthermore, extending this architecture to support simultaneous multi-frame decoding of different block lengths and code rates could also be investigated.
3. The two multi-frame decoding approaches were introduced in the purpose of increasing the throughput and the hardware efficiency of the decoder. Moreover, such an approach is very useful for the blind decoding of the PDCCH polar codes [23, 69].

Indeed, 44 PDCCH candidates are decoded at the user equipment (UE) to identify the downlink control information (DCI) carrying the UE control information [1]. However, at the receiver, the UE does not need to decode the PDCCH candidate that belongs to the search space with the list-augmented SC algorithm. Therefore, a research study item would be to investigate the use of the proposed list-augmented decoder ($L = 8$) as multiple SC decoder cores (L-SC) in order to reduce the search space of candidates as a first phase of blind decoding [24]. Then, the list-augmented SC decoder can be used for decoding a reduced number of candidates in the second phase of decoding.

Bibliography

[1] 3GPP, "Physical layer procedures," document 3GPP TS 36.213 V.8.2.0 (2018-06), 3rd Generation Partnership Project (3GPP), 2008. (Cit. on pp. xiii, 108).

[2] Technical specification group radio access network; study on scenarios and requirements for next generation access technologies, release 15, v15.0.0," 3GPP, Valbonne, France, 3GPP Rep. TR 38.913, May 2018. (Cit. on p. 3).

[3] 3GPP, "NR; Multiplexing and Channel Coding (Release 15)," Tech. Rep. TS 38.212V15.2.0 (2018-06), Jan. 2018. [Online]. Available: `https://www.3gpp.org/DynaReport/38-series.htm` (cit. on pp. 12, 34, 39, 40).

[4] Orion Afisiadis, Alexios Balatsoukas-Stimming, and Andreas Burg, « A low-complexity improved successive cancellation decoder for polar codes », *2014 48th Asilomar Conference on Signals, Systems and Computers*, IEEE, 2014, pp. 2116–2120 (cit. on pp. 15, 23).

[5] Amin Alamdar-Yazdi and Frank R Kschischang, « A simplified successive-cancellation decoder for polar codes », *IEEE communications letters* 15.12 (2011), pp. 1378–1380 (cit. on p. 25).

[6] NGMN Alliance, « 5G white paper », *Next generation mobile networks, white paper* 1.2015 (2015) (cit. on p. 3).

[7] Maryam Haghighi Ardakani et al., « Fast successive-cancellation-based decoders of polar codes », *IEEE Transactions on Communications* 67.7 (2019), pp. 4562–4574 (cit. on p. 32).

[8] Erdal Arikan, « A performance comparison of polar codes and Reed-Muller codes », *IEEE Communications Letters* 12.6 (2008), pp. 447–449 (cit. on p. 16).

[9] Erdal Arikan, « Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels », *IEEE Transactions on information Theory* 55.7 (2009), pp. 3051–3073 (cit. on pp. 12, 13, 15, 16).

[10] Erdal Arikan, « Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels », *IEEE Transactions on information Theory* 55.7 (2009), pp. 3051–3073 (cit. on pp. 12, 22).

[11] Erdal Arıkan, « Polar codes: A pipelined implementation », *Proc. 4th ISBC* (2010), pp. 11–14 (cit. on pp. 15, 21).

[12] Jung Hyun Bae et al., « An overview of channel coding for 5G NR cellular communications », *APSIPA Transactions on Signal and Information Processing* 8 (2019) (cit. on p. 3).

[13] Alexios Balatsoukas-Stimming, Mani Bastani Parizi, and Andreas Burg, « LLR-Based Successive Cancellation List Decoding of Polar Codes », *IEEE Transactions on Signal Processing* 63.19 (2015), pp. 5165–5179, DOI: 10.1109/TSP.2015.2439211 (cit. on p. 43).

[14] Alexios Balatsoukas-Stimming, Mani Bastani Parizi, and Andreas Burg, « LLR-based successive cancellation list decoding of polar codes », *IEEE transactions on signal processing* 63.19 (2015), pp. 5165–5179 (cit. on pp. 46, 50).

[15] Alexios Balatsoukas-Stimming, Mani Bastani Parizi, and Andreas Burg, « On metric sorting for successive cancellation list decoding of polar codes », *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2015, pp. 1993–1996 (cit. on p. 20).

[16] Alexios Balatsoukas-Stimming et al., « Hardware Architecture for List Successive Cancellation Decoding of Polar Codes », *IEEE Transactions on Circuits and Systems II: Express Briefs* 61.8 (2014), pp. 609–613, DOI: 10.1109/TCSII.2014.2327336 (cit. on p. 42).

[17] Valerio Bioglio, Carlo Condo, and Ingmar Land, « Design of polar codes in 5G new radio », *IEEE Communications Surveys & Tutorials* (2020) (cit. on p. 35).

[18] Valerio Bioglio et al., « Two-step metric sorting for parallel successive cancellation list decoding of polar codes », *IEEE Communications Letters* 21.3 (2016), pp. 456–459 (cit. on p. 20).

[19] Ludovic Chandesris, Valentin Savin, and David Declercq, « An improved SCFlip decoder for polar codes », *2016 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2016, pp. 1–6 (cit. on p. 23).

[20] Fengyi Cheng et al., « Bit-flip algorithm for successive cancellation list decoder of polar codes », *IEEE Access* 7 (2019), pp. 58346–58352 (cit. on p. 23).

[21] Lalit Chettri and Rabindranath Bera, « A comprehensive survey on Internet of Things (IoT) toward 5G wireless systems », *IEEE Internet of Things Journal* 7.1 (2019), pp. 16–32 (cit. on p. 3).

[22] Carlo Condo, Valerio Bioglio, and Ingmar Land, « Generalized fast decoding of polar codes », *2018 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2018, pp. 1–6 (cit. on pp. xiii, 28, 107).

[23] Carlo Condo, Seyyed Ali Hashemi, and Warren J Gross, « Blind detection with polar codes », *IEEE Communications Letters* 21.12 (2017), pp. 2550–2553 (cit. on pp. xiii, 34, 107).

[24] Carlo Condo et al., « Design and implementation of a polar codes blind detection scheme », *IEEE Transactions on Circuits and Systems II: Express Briefs* 66.6 (2018), pp. 943–947 (cit. on pp. xiv, 108).

[25] Bin Dai et al., « Parity Check Aided SC-Flip Decoding Algorithms for Polar Codes », *IEEE Transactions on Vehicular Technology* 70.10 (2021), pp. 10359–10368 (cit. on p. 23).

[26] Nghia Doan, Seyyed Ali Hashemi, and Warren J Gross, « Fast Successive-Cancellation List Flip Decoding of Polar Codes », *IEEE Access* (2022) (cit. on p. 26).

[27] Zeynep B Kaykac Egilmez et al., « The development, operation and performance of the 5G polar codes », *IEEE Communications Surveys & Tutorials* 22.1 (2019), pp. 96–122 (cit. on pp. vii, 4).

[28] Furkan Ercan, Thibaud Tonnellier, and Warren J Gross, « Energy-efficient hardware architectures for fast polar decoders », *IEEE Transactions on Circuits and Systems I: Regular Papers* 67.1 (2019), pp. 322–335 (cit. on p. 46).

[29] Furkan Ercan et al., « On error-correction performance and implementation of polar code list decoders for 5G », *2017 55th annual allerton conference on communication, control, and computing (allerton)*, IEEE, 2017, pp. 443–449 (cit. on p. 21).

[30] Furkan Ercan et al., « Partitioned successive-cancellation flip decoding of polar codes », *2018 IEEE International Conference on Communications (ICC)*, IEEE, 2018, pp. 1–6 (cit. on p. 23).

[31] Furkan Ercan et al., « Practical dynamic SC-flip polar decoders: Algorithm and implementation », *IEEE Transactions on Signal Processing* 68 (2020), pp. 5441–5456 (cit. on p. 23).

[32] YouZhe Fan and Chi-ying Tsui, « An efficient partial-sum network architecture for semi-parallel polar codes decoder implementation », *IEEE Transactions on Signal Processing* 62.12 (2014), pp. 3165–3179 (cit. on pp. 89, 90).

[33] Ubaid U Fayyaz and John R Barry, « Low-complexity soft-output decoding of polar codes », *IEEE Journal on Selected Areas in Communications* 32.5 (2014), pp. 958–966 (cit. on p. 23).

[34] Gerhard P. Fettweis, « The Tactile Internet: Applications and Challenges », *IEEE Veh. Technol. Mag.* 9.1 (2014), pp. 64–70 (cit. on p. 97).

[35] Robert Gallager, « Low-density parity-check codes », *IRE Transactions on information theory* 8.1 (1962), pp. 21–28 (cit. on p. 12).

[36] Pascal Giard and Andreas Burg, « Fast-SSC-flip decoding of polar codes », *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, IEEE, 2018, pp. 73–77 (cit. on p. 26).

[37] Pascal Giard et al., « 237 Gbit/s unrolled hardware polar decoder », *Electronics Letters* 51.10 (2015), pp. 762–763 (cit. on pp. 46, 47).

[38] Pascal Giard et al., « Multi-mode unrolled architectures for polar decoders », *IEEE Transactions on Circuits and Systems I: Regular Papers* 63.9 (2016), pp. 1443–1453 (cit. on p. 47).

[39] Fatemeh Hamidi-Sepehr, Ajit Nimbalker, and Gregory Ermolaev, « Analysis of 5G LDPC codes rate-matching design », *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, IEEE, 2018, pp. 1–5 (cit. on p. 4).

[40] Muhammad Hanif, Maryam H Ardakani, and Masoud Ardakani, « Fast list decoding of polar codes: Decoders for additional nodes », *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, IEEE, 2018, pp. 37–42 (cit. on pp. xiii, 32, 33, 107).

[41] Muhammad Hanif and Masoud Ardakani, « Fast successive-cancellation decoding of polar codes: Identification and decoding of new nodes », *IEEE Communications Letters* 21.11 (2017), pp. 2360–2363 (cit. on p. 27).

[42] Seyyed Ali Hashemi, Carlo Condo, and Warren J Gross, « A fast polar code list decoder architecture based on sphere decoding », *IEEE Transactions on Circuits and Systems I: Regular Papers* 63.12 (2016), pp. 2368–2380 (cit. on pp. 28, 30, 32).

[43] Seyyed Ali Hashemi, Carlo Condo, and Warren J Gross, « Fast and flexible successive-cancellation list decoders for polar codes », *IEEE Transactions on Signal Processing* 65.21 (2017), pp. 5756–5769 (cit. on pp. 31, 32, 42, 46, 52).

[44] Seyyed Ali Hashemi, Carlo Condo, and Warren J Gross, « List sphere decoding of polar codes », *2015 49th Asilomar Conference on Signals, Systems and Computers*, IEEE, 2015, pp. 1346–1350 (cit. on p. 29).

[45] Seyyed Ali Hashemi, Carlo Condo, and Warren J Gross, « Simplified successive-cancellation list decoding of polar codes », *2016 IEEE International Symposium on Information Theory (ISIT)*, IEEE, 2016, pp. 815–819 (cit. on pp. 26, 29, 30, 32).

[46] Seyyed Ali Hashemi, Carlo Condo, and Warren J. Gross, « Simplified Successive-Cancellation List decoding of polar codes », *IEEE International Symposium on Information Theory (ISIT)*, 2016, pp. 815–819, DOI: 10.1109/ISIT.2016.7541412 (cit. on p. 29).

[47] Seyyed Ali Hashemi et al., « On the performance of polar codes for 5G eMBB control channel », *2017 51st Asilomar Conference on Signals, Systems, and Computers*, IEEE, 2017, pp. 1764–1768 (cit. on p. 4).

[48] Seyyed Ali Hashemi et al., « Partitioned successive-cancellation list decoding of polar codes », *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Ieee, 2016, pp. 957–960 (cit. on p. 21).

[49] Seyyed Ali Hashemi et al., « Rate-flexible fast polar decoders », *IEEE Transactions on Signal Processing* 67.22 (2019), pp. 5689–5701 (cit. on p. 92).

[50] Jui-Hui Hung and Sau-Gee Chen, « A 1.45 Gb/s (576,288) LDPC decoder for 802.16 e standard », *2007 IEEE International Symposium on Signal Processing and Information Technology*, IEEE, 2007, pp. 916–921 (cit. on p. 21).

[51] Nadine Hussami, Satish Babu Korada, and Rudiger Urbanke, « Performance of polar codes for channel and source coding », *2009 IEEE International Symposium on Information Theory*, IEEE, 2009, pp. 1488–1492 (cit. on p. 15).

[52] Byeong Yong Kong, Hoyoung Yoo, and In-Cheol Park, « Efficient sorting architecture for successive-cancellation-list decoding of polar codes », *IEEE Transactions on Circuits and Systems II: Express Briefs* 63.7 (2016), pp. 673–677 (cit. on p. 20).

[53] Satish Babu Korada, Eren Şaşoğlu, and Rüdiger Urbanke, « Polar codes: Characterization of exponent, bounds, and constructions », *IEEE Transactions on Information Theory* 56.12 (2010), pp. 6253–6264 (cit. on p. 40).

[54] Bertrand Le Gal et al., « Low-latency sorter architecture for polar codes successive-cancellation-list decoding », *2020 IEEE Workshop on Signal Processing Systems (SiPS)*, IEEE, 2020, pp. 1–5 (cit. on p. 20).

[55] Camille Leroux et al., « A Semi-Parallel Successive-Cancellation Decoder for Polar Codes », *IEEE Transactions on Signal Processing* 61.2 (2013), pp. 289–299, DOI: 10.1109/TSP.2012.2223693 (cit. on pp. 6, 48, 50, 57, 58, 66, 70, 80, 89).

[56] Camille Leroux et al., « Hardware architectures for successive cancellation decoding of polar codes », *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2011, pp. 1665–1668 (cit. on p. 70).

[57] Bin Li, Hui Shen, and David Tse, « An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check », *IEEE communications letters* 16.12 (2012), pp. 2044–2047 (cit. on p. 21).

[58] Huan Li, « Enhanced metric sorting for successive cancellation list decoding of polar codes », *IEEE Communications Letters* 22.4 (2018), pp. 664–667 (cit. on p. 20).

[59] Huijun Li and Jinhong Yuan, « A practical construction method for polar codes in AWGN channels », *IEEE 2013 Tencon-Spring*, IEEE, 2013, pp. 223–226 (cit. on p. 40).

[60] Xiao Liang et al., « Efficient stochastic successive cancellation list decoder for polar codes », *Science China Information Sciences* 63.10 (2020), pp. 1–19 (cit. on p. 99).

[61] Xiao Liang et al., « Hardware efficient and low-latency CA-SCL decoder based on distributed sorting », *2016 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2016, pp. 1–6 (cit. on pp. 98, 99).

[62] Jun Lin, Chenrong Xiong, and Zhiyuan Yan, « A high throughput list decoder architecture for polar codes », *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 24.6 (2015), pp. 2378–2391 (cit. on p. 43).

[63] Jun Lin and Zhiyuan Yan, « A hybrid partial sum computation unit architecture for list decoders of polar codes », *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2015, pp. 1076–1080 (cit. on p. 89).

[64] Chih-Hao Liu et al., « An LDPC decoder chip based on self-routing network for IEEE 802.16 e applications », *IEEE Journal of Solid-State Circuits* 43.3 (2008), pp. 684–694 (cit. on p. 21).

[65] Chih-Hao Liu et al., « Design of a multimode QC-LDPC decoder based on shift-routing network », *IEEE Transactions on Circuits and Systems II: Express Briefs* 56.9 (2009), pp. 734–738 (cit. on p. 21).

[66] Xiaocheng Liu et al., « A 5.16 Gbps decoder ASIC for polar code in 16nm FinFET », *2018 15th International Symposium on Wireless Communication Systems (ISWCS)*, IEEE, 2018, pp. 1–5 (cit. on p. 46).

[67] Robert G Maunder, « The 5G channel code contenders », *AccelerComm white paper* (2016), pp. 1–13 (cit. on p. 97).

[68] Anadi Mishra et al., « A successive cancellation decoder ASIC for a 1024-bit polar code in 180nm CMOS », *2012 IEEE Asian solid state circuits conference (A-SSCC)*, IEEE, 2012, pp. 205–208 (cit. on p. 46).

[69] Reza Moosavi and Erik G Larsson, « A fast scheme for blind identification of channel codes », *2011 IEEE Global Telecommunications Conference-GLOBECOM 2011*, IEEE, 2011, pp. 1–5 (cit. on pp. xiii, 107).

[70] Mahsa Mousavi et al., « Efficient partial-sum network architectures for list successive-cancellation decoding of polar codes », *IEEE Transactions on Signal Processing* 66.14 (2018), pp. 3848–3858 (cit. on pp. 89, 90).

[71] Seho Myung, Kyeongcheol Yang, and Jaeyoel Kim, « Quasi-cyclic LDPC codes for fast encoding », *IEEE Transactions on Information Theory* 51.8 (2005), pp. 2894–2901 (cit. on p. 12).

[72] Kai Niu and Kai Chen, « CRC-aided decoding of polar codes », *IEEE communications letters* 16.10 (2012), pp. 1668–1671 (cit. on pp. 20, 21).

[73] Kai Niu and Kai Chen, « Stack decoding of polar codes », *Electronics letters* 48.12 (2012), pp. 695–697 (cit. on pp. 15, 23).

[74] Klaus Pedersen and Troels Kolding, « Overview of 3GPP New Radio Industrial IoT Solutions », *Wireless Networks and Industrial IoT: Applications, Challenges and Enablers*, ed. by Nurul Huda Mahmood et al., Cham: Springer International Publishing, 2021, pp. 3–20, ISBN: 978-3-030-51473-0, DOI: 10.1007/978-3-030-51473-0_1, URL: https://doi.org/10.1007/978-3-030-51473-0_1 (cit. on p. 3).

[75] Charles Pillet, Valerio Bioglio, and Carlo Condo, « On list decoding of 5G-NR polar codes », *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, IEEE, 2020, pp. 1–6 (cit. on p. 4).

[76] Charles Pillet, Carlo Condo, and Valerio Bioglio, « SCAN list decoding of polar codes », *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, IEEE, 2020, pp. 1–6 (cit. on p. 23).

[77] Mohammad Rowshan et al., « Logarithmic Non-uniform Quantization for List Decoding of Polar Codes », *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, IEEE, 2021, pp. 1161–1166 (cit. on p. 43).

[78] Gabi Sarkis et al., « Fast list decoders for polar codes », *IEEE Journal on Selected Areas in Communications* 34.2 (2015), pp. 318–328 (cit. on p. 21).

[79] Gabi Sarkis et al., « Fast Polar Decoders: Algorithm and Implementation », *IEEE Journal on Selected Areas in Communications* 32.5 (2014), pp. 946–957, DOI: 10.1109/JSAC.2014.140514 (cit. on p. 25).

[80] Kinza Shafique et al., « Internet of things (IoT) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5G-IoT scenarios », *Ieee Access* 8 (2020), pp. 23022–23040 (cit. on p. 3).

[81] Claude Elwood Shannon, « A mathematical theory of communication », *The Bell system technical journal* 27.3 (1948), pp. 379–423 (cit. on pp. 13, 39).

[82] Yifei Shen et al., « Enhanced belief propagation decoder for 5G polar codes with bit-flipping », *IEEE Transactions on Circuits and Systems II: Express Briefs* 67.5 (2020), pp. 901–905 (cit. on p. 23).

[83] Zukang Shen et al., « Overview of 3GPP LTE-advanced carrier aggregation for 4G wireless communications », *IEEE Communications Magazine* 50.2 (2012), pp. 122–130 (cit. on p. 3).

[84] Wenqing Song et al., « Efficient successive cancellation stack decoder for polar codes », *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 27.11 (2019), pp. 2608–2619 (cit. on p. 23).

[85] Ido Tal and Alexander Vardy, « How to construct polar codes », *IEEE Transactions on Information Theory* 59.10 (2013), pp. 6562–6582 (cit. on p. 40).

[86] Ido Tal and Alexander Vardy, « List decoding of polar codes », *2011 IEEE International Symposium on Information Theory Proceedings*, 2011, pp. 1–5, DOI: 10.1109/ISIT.2011.6033904 (cit. on p. 20).

[87] Ido Tal and Alexander Vardy, « List decoding of polar codes », *IEEE Transactions on Information Theory* 61.5 (2015), pp. 2213–2226 (cit. on pp. 15, 18, 20).

[88] TechSpot, *Everything you need to know about 4g wireless technology*, TechSpot https://www.techspot.com/guides/272-everything-about-4g/. 2010 (cit. on p. 3).

[89] Harish Vangala, Emanuele Viterbo, and Yi Hong, « A comparative study of polar code constructions for the AWGN channel », *arXiv preprint arXiv:1501.02473* (2015) (cit. on p. 40).

[90] Tao Wang, Daiming Qu, and Tao Jiang, « Parity-check-concatenated polar codes », *IEEE Communications Letters* 20.12 (2016), pp. 2342–2345 (cit. on p. 35).

[91] Xiumin Wang et al., « Improved metric sorting for successive cancellation list decoding of polar codes », *IEEE Communications Letters* 23.7 (2019), pp. 1123–1126 (cit. on p. 20).

[92] Daolong Wu, Ying Li, and Yue Sun, « Construction and block error rate analysis of polar codes over AWGN channel based on Gaussian approximation », *IEEE Communications Letters* 18.7 (2014), pp. 1099–1102 (cit. on p. 40).

[93] ChenYang Xia et al., « An implementation of list successive cancellation decoder with large list size for polar codes », *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, IEEE, 2017, pp. 1–4 (cit. on p. 46).

[94] Luping Xiang et al., « CRC-aided logarithmic stack decoding of polar codes for ultra reliable low latency communication in 3GPP new radio », *IEEE Access* 7 (2019), pp. 28559–28573 (cit. on p. 24).

[95] *Xilinx, Intelectual property Polar Encoder/Decoder*, `https://www.xilinx.com/products/intellectual-property/ef-di-polar-enc-dec.html#overview`, Reno, USA, November 2016 (cit. on pp. 98, 99).

[96] Chenrong Xiong et al., « An FPGA emulation platform for polar codes », *2016 IEEE International Workshop on Signal Processing Systems (SiPS)*, IEEE, 2016, pp. 148–153 (cit. on p. 99).

[97] Yu Yongrun et al., « Successive cancellation list bit-flip decoder for polar codes », *2018 10th International Conference on Wireless Communications and Signal Processing (WCSP)*, IEEE, 2018, pp. 1–6 (cit. on p. 23).

[98] Yongrun Yu et al., « Belief propagation bit-flip decoder for polar codes », *IEEE Access* 7 (2019), pp. 10937–10946 (cit. on p. 23).

[99] Bo Yuan and Keshab K Parhi, « Low-latency successive-cancellation polar decoder architectures using 2-bit decoding », *IEEE Transactions on Circuits and Systems I: Regular Papers* 61.4 (2013), pp. 1241–1254 (cit. on p. 46).

[100] Chuan Zhang and Keshab K Parhi, « Low-latency sequential and overlapped architectures for successive cancellation polar decoder », *IEEE Transactions on Signal Processing* 61.10 (2013), pp. 2429–2441 (cit. on p. 89).

[101] Si-yu Zhang and Behnam Shahrrava, « Enhanced BP decoding schemes of polar codes », *IET Communications* 15.9 (2021), pp. 1133–1142 (cit. on p. 21).

[102] Yangcan Zhou, Jun Lin, and Zhongfeng Wang, « A new fast-ssc-flip decoding of polar codes », *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, IEEE, 2019, pp. 1–6 (cit. on p. 26).

[103] Yangcan Zhou, Jun Lin, and Zhongfeng Wang, « Improved fast-SSC-flip decoding of polar codes », *IEEE Communications Letters* 23.6 (2019), pp. 950–953 (cit. on p. 23).

DOCTORAT BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Nouveaux défis dans la conception de décodeurs de codes polaires pour la 5G.

Mot clés : Codes correcteurs d'erreurs, codes polaires, décodage à annulation successive, décodage à liste, faible latence, implémentation matérielle, flexibilité, FPGA.

Résumé : Proposés ces dernières années, les codes polaires représentent l'un des derniers apports à la famille des codes correcteurs d'erreurs (FEC). Ils ont été adoptés comme schéma de codage pour le canal de contrôle de la norme 5G NR (New Radio) pour la cinquième génération de communications mobiles cellulaires. Cependant, les exigences élevées introduites par le canal de contrôle de la 5G en termes de flexibilité de la longueur de code et du rendement de codage font que la plupart des décodeurs matériels de codes polaires publiés antérieurement sont inadaptés. En effet, ces derniers se sont principalement focalisés sur des décodeurs à annulation successive offrant un débit élevé, une flexibilité limitée et des pouvoirs de correction d'erreurs réduits. Avec des contraintes strictes sur le délai de bout en bout et sur la correction d'erreurs, la 5G NR nécessite des architectures de décodeurs à liste à faible latence. Dans ce contexte, plusieurs contributions originales sont proposées dans ce travail de thèse. La première contribution majeure est liée à l'exploration de l'espace de conception et concerne l'étude de l'impact des principaux paramètres du code et du décodeur sur la latence, le débit et la complexité matérielle des architectures de décodage semi-parallèles. L'impact de ces paramètres sur l'efficacité matérielle des architectures semi-parallèles est important. Par conséquent, nous proposons deux approches de décodage multi-trames qui augmentent le débit et améliorent le taux d'utilisation des unités de traitement de ces architectures. Des résultats analytiques détaillés et des résultats de synthèse logique sont fournis et comparés pour une large gamme de valeurs afin de constituer une référence pour la mise en œuvre de décodeurs FEC flexibles mais efficaces pour les codes polaires. Par ailleurs, un environnement logiciel complet de simulation du codage/décodage de codes polaires est proposé pour évaluer les performances de différents algorithmes dans une représentation des données en virgule flottante et en virgule fixe. La deuxième contribution majeure concerne la conception d'une architecture matérielle originale, flexible et à faible latence, basée sur le décodage à liste des codes polaires de la 5G NR. Ce résultat a été obtenu sur la base de l'étude d'exploration de l'espace de conception, et motivé par le besoin de fournir un décodeur polaire efficace qui supporte les niveaux de flexibilité et de latence requis pour le standard. Le décodeur proposé supporte toutes les tailles de trame et tous les rendements de code définis dans la 5G avec des valeurs de débit et de latence conformes aux exigences de la norme. En outre, l'architecture du décodeur a été étendue pour supporter le schéma de décodage multi-trame proposé, particulièrement adapté au décodage aveugle des informations de contrôle de la liaison descendante.

Title: New challenges in designing polar code decoders for 5G

Keywords: Forward error correction, polar codes, successive cancellation decoding, list decoding, low latency, hardware design, flexibility, FPGA.

Abstract: Proposed in the last few years, polar codes represent one of the latest additions to the family of forward error correction (FEC) codes. They have been adopted as the coding scheme in the control channel of the 3rd Generation Partnership Project (3GPP) New Radio (NR) standard for the fifth generation of cellular mobile communications (5G). However, the challenging requirements introduced by the 5G control channel in terms of block length and code rate flexibility render unsuitable most of the previously published hardware polar decoder implementations. Indeed, these latter focused mainly on successive cancellation decoders with high throughput, limited flexibility and error correction capabilities. With stringent constraints on end-to-end delay and error correction, the 5G NR steers towards low-latency list-based decoder architectures. In this context, several original contributions are proposed in this thesis work. The first major contribution is related to design space exploration and concerns the study of the impact of main code and decoder design parameters on the latency, throughput, and the hardware complexity of semi-parallel decoding architectures. The impact of these parameters on the hardware efficiency of semi-parallel architectures is significant. Therefore, we propose two multi-frame decoding approaches that increase the throughput and improve the utilisation rate of the processing units of these architectures. Detailed analytical and logic synthesis results are provided and compared for a large range of values in order to constitute a reference for the implementation of flexible, yet efficient FEC decoders for polar codes. Furthermore, a complete software simulation environment of polar coding/decoding is proposed for performance evaluation under different algorithms in both floating-point and fixed-point data representation. The second major contribution concerns the design of an original flexible and low-latency list-based hardware architecture for decoding 5G NR polar codes. This was achieved based on the design space exploration study, and motivated by the need to provide a hardware-efficient polar decoder that supports the required flexibility and latency levels for 5G NR. The proposed design supports all the frame sizes and code rates defined in 3GPP with throughput and latency values meeting the standard requirements. Furthermore, the decoder architecture has been extended to support the proposed multi-frame decoding scheme, particularly suited for blind decoding of downlink control information.