POLITECNICO DI TORINO

III Facoltà di Ingegneria
Corso di Laurea in Ingegneria delle Telecomunicazioni

Tesi di Laurea Magistrale

BCH-LDPC Concatenated Coding and High Order Modulations for Satellite Transmitters

Relatore:
prof. Roberto Garello

Candidato:
Eriprando Pacces

Supervisore aziendale
Thales Alenia Space Italia
ing. Domenico Giancristofaro

Luglio 2008

Sommario

Il lavoro svolto nell'ambito di questa tesi ha avuto l'obiettivo di consentire la progettazione di un dispositivo di trasmissione di bordo per comunicazioni via satellite, fondato sullo standard DVB-S2, contribuendo ad una particolare sezione del progetto. Si sono analizzate e impiegate le più moderne tecniche nel campo della modulazione e codifica, viste le prestazioni altamente sfidanti da raggiungere nel particolare scenario applicativo.

In ogni sistema di comunicazione la scelta e il progetto degli schemi di modulazione e codifica ha come finalità quella di trasmettere i dati in maniera sufficientemente affidabile per il tipo di applicazione richiesta, facendo al tempo stesso un uso efficiente delle risorse disponibili (banda, potenza, e, in particolare nelle comunicazioni via satellite, peso ed ingombro delle apparecchiature).

Nel 1948, Claude E. Shannon definisce e quantifica l'informazione, riuscendo inoltre a dimostrare che, per una qualsivoglia velocità di trasmissione minore o uguale a un cardinale parametro chiamato capacità di canale, esiste uno schema di codifica in grado di ottenere una probabilità d'errore arbitrariamente piccola, garantendo così una trasmissione completamente affidabile dei dati. Purtroppo Shannon nel suo teorema non dà alcuna indicazione su come poter trovare questi schemi di codifica. Shannon dimostra tuttavia che codici lunghi scelti casualmente assicurano una bassa probabilità d'errore media. Purtroppo una diretta realizzazione dei codici suddetti porta ad avere una complessità di decodifica molto alta. Dal 1948, quindi, l'impegno degli ingegneri delle telecomunicazioni si concentra sullo sviluppo di schemi realizzabili di modulazione e codifica, che si avvicinino alle prestazioni limite. Nonostante l'iniziale pessimismo (si pensi al motto *"good codes are messy"* che circolava tra gli studiosi di teoria dei codici), il problema viene risolto almeno per il caso di maggior interesse e importanza, il canale lineare gaussiano bianco (AWGN, *Additive White Gaussian Noise*).

Nel 1980, poi, la migliore comprensione del significato del teorema di Shannon porta a una revisione del paradigma utilizzato sino ad allora nel campo delle modulazioni e della codifica. Queste due discipline, che prima di questo momento si sviluppavano indipendentemente, incominciano a diventare rigidamente collegate.

Parallelamente, l'esigenza di ottenere alte efficienze spettrali fa aumentare la cardinalità delle modulazioni (il numero delle forma d'onda impiegate in ogni simbolo); nel frattempo, vengono ideati codici a correzione d'errore ancora più efficaci. In tale contesto il lavoro di Ungerboek sancisce l'arrivo dei codici TCM (*Trellis Coded Modulation*), rendendo chiari i vantaggi del trattare modulazione e codifica come una entità singola. Inoltre, vari schemi pubblicati in letteratura dimostrano che turbo codici (una tecnica di codifica introdotta nel 1993 da Claude Berrou) e TCM possono essere utilizzati con profitto congiuntamente. Tuttavia i turbo codici dovrebbero essere progettati ad hoc per ogni singola modulazione, una soluzione impraticabile per sistemi che adottano diverse costellazioni. La soluzione di questo problema prende il nome di *Bit Interleaved Coding Modulation* (BICM), tecnica grazie alla quale è possibile ottenere dei risultati eccellenti impiegando una singola soluzione di codifica per tutte le modulazioni previste. Questa grande flessibilità rende le BICM commercialmente appetibili per sistemi comunicazione satellitari a banda larga di tipo adattativo (*Adaptive Coding Modulation*, ACM) che necessitano di un ampio ventaglio di efficienze spettrali. Appartengono alle soluzioni ACM anche lo standard ETSI DVB-S2 e la soluzione MHOMS in via di standardizzazione nel CCSDS (*Consultive Committee for Space Data Systems*).

Il recente standard DVB-S2, successivo al già consolidato DVB-S, utilizza gran parte delle più recenti e innovative tecniche nel campo della codifica di canale e delle modulazioni congiuntamente a modulazioni ad alta cardinalità, come la 16APSK e la 32APSK. Il 'motore' (ovvero l'elemento essenziale) della codifica di canale è composto da un codice LDPC concatenato a un BCH per limitarne il tipico calo delle prestazioni ad altri rapporti segnali rumore, che prende il nome di *error floor*. Questa scelta garantisce delle prestazioni (riferendoci alla sola modulazione e codifica) tra 0,6-1,2 dB dal limite di Shannon. Oltre a ciò, il DVB-S2, offrendo un gran numero di modalità operative e quindi un ampio raggio di efficienze di banda per la trasmissione del segnale, è adatto a numerose tipologie di missione spaziale. Le caratteristiche di alta regolarità e periodicità dell'LDPC, poi, garantiscono una complessità ragionevole sia in fase di codifica che in fase di decodifica.

Dopo una parte introduttiva dedicata allo studio delle principali caratteristiche ed applicazioni delle comunicazioni satellitari, quali i servizi di diffusione del segnale televisivo, telerilevamento, raccolta dei dati, etc., si descrivono i blocchi funzionali della sezione di trasmissione del DVB-S2, che come già accennato, raccoglie gran parte degli aspetti più innovativi nel campo delle modulazioni e codifica di canale. Si mette inoltre in evidenza come la tecnica ACM, ossia di modulazione e codifica adattativa, possa garantire un più alto *throughput*, una maggiore disponibilità del servizio (specie se confrontata con quella ottenuta adoperando modulazione e codifica costanti (CCM)), una maggiore efficienza e ottimizzazione delle risorse disponibili a bordo del satellite. Si passa quindi alla revisione critica della codifica di canale

(FEC, *Forward Error Correction*) del DVB-S2. Dopo una breve descrizione dei principali criteri di costruzione dei codici BCH, vengono illustrate le caratteristiche del particolare codice BCH adoperato nello standard DVB-S2, per poi passare alla descrizione dei codici LDPC; cercando di dare una giustificazione della loro "rinascita", nonché di spiegarne la struttura, i metodi e gli algoritmi di codifica, e specialmente la loro decodifica iterativa a "passaggio di messaggi".

La parte centrale della tesi è invece dedicata allo studio di algoritmi di codifica efficienti e veloci dei codici BCH e alla loro realizzazione in hardware. Dopo una fase preliminare di analisi del tradizionale algoritmo di codifica per il codice BCH sistematico (quale quello utilizzato dal DVB-S2), si descrivono le possibili architetture seriali capaci di realizzare il metodo di codifica, analizzandone pregi e difetti in vista della successiva realizzazione in hardware. Queste semplici architetture sono intrinsecamente seriali e quindi lente; esse sono chiamate Linear Feedback Shift Register (LFSR) e realizzano la divisione per il polinomio generatore del codice. Tale metodo di codifica è tipico della famiglia dei codici ciclici a cui appartengono i codici BCH. Dal punto di vista hardware, la lunghezza delle parole di codice (previste dal DVB-S2), i vincoli di frequenza imposti dell'analisi globale della sezione di trasmissione e la definizione dell'interfaccia con il codificatore LDPC che ha un ingresso necessariamente parallelo, impongono lo sviluppo di un modello di codificatore BCH a sezioni multiple che risponda alle specifiche di velocità e sia agevolmente integrabile con l'intera sezione di trasmissione del DVB-S2. La descrizione matematica della versione dell'algoritmo di codifica parallelo (ottenuta per mezzo delle equazioni di stato) e, in particolare, le fortunate proprietà di regolarità della matrice di transizione di stato $\mathbf{A}$ ricavata dal modello del sistema conducono ad una realizzazione particolarmente efficiente del codificatore BCH. Oltretutto, l'algebra binaria usata nelle operazioni di codifica, per cui le operazioni di somma vengono direttamente realizzate per mezzo di porte XOR mentre quelle di moltiplicazione per mezzo di AND, riduce drasticamente il carico computazionale e, specialmente, il contributo di ritardo delle operazioni elementari ai cammini critici del grafo della elaborazione algoritmica per la frequenza di clock da raggiungersi.

La scelta dell'architettura hardware e del suo livello di parallelismo (scelta pari a 8 rami paralleli, sebbene il modello matematico sviluppato sia del tutto generale e consenta di modificare a piacimento tale parametro progettuale) è stata dettata, come già detto, non solo dai vincoli imposti dalla sezione di trasmissione realizzata, ma specialmente dalla esigenza di disporre di un codificatore in grado di operare in modalità ACM, ossia in accordo con le raccomandazioni dello standard ETSI DVB-S2, secondo il quale le modalità di modulazioni e codifica possono variare da una trama all'altra. Ciò comporta anche la memorizzazione di alcuni coefficienti di tutte le matrici $\mathbf{A}^p$ e $\mathbf{B}_p$ (dove $\mathbf{B}_p$ è la matrice trasferimento ingresso-stato) associate alle varie capacità correttive del codice BCH in delle tabelle di riferimento o *Look Up Table* (LUT). L'architettura hardware realizzata in TAS-I comprende un

registro a 192 celle (*Flip Flop*), 192 reti combinatorie d'ingresso e 192 reti combinatorie d'uscita, pilotate opportunamente dai coefficienti (precedentemente calcolati) memorizzati nelle LUT, e alcuni sommatori, che, come già osservato, sono delle porte XOR a due ingressi. Le reti combinatorie sono in realtà delle porte XOR a otto ingressi, la cui abilitazione è pilotata (per mezzo di porte AND a 2 ingressi) dai coefficienti precalcolati e memorizzati nelle LUT. Il poter pilotare le reti combinatorie per mezzo di alcune LUT rende possibile la codifica, lasciando ovviamente intatta l'architettura ideata, per tutte le modalità operative previste (modulazione e codifica BCH-LDPC) dallo standard ETSI DVB-S2. Si tratta semplicemente di inibire le prime 32 o 64 coppie di reti combinatorie (quelle che operano sugli ingressi e quelle che agiscono sul bus di uscita) a seconda della capacità correttiva del codice BCH, ossia 10 e 8.

Un capitolo della tesi è dedicato alla descrizione del lavoro di simulazione svolto in linguaggio C/C++ delle architetture e degli algoritmi impiegati allo scopo di fornire supporto e dati di riferimento per la di verifica funzionale ai progettisti VLSI (*Very Large Scale Integration*) nella sintesi del progetto mediante linguaggio di progettazione digitale VHDL (*Very high speed integrated circuits Hardware Description Language*), e nella successiva verifica dei risultati prodotti dal modello descritto. Tutto questo, però, non prima di aver verificato mediante un adeguato numero di simulazioni in linguaggio C la corretta generazione delle parole di codice (tramite un programma di rivelazione degli errori) nonché l'effettiva capacità correttiva del codice, attraverso l'inserzione di errori pseudocasuali e successiva elaborazione di decodifica. Una parte del capitolo è poi dedicata all'algoritmo di decodifica impiegato, quello di Berlekamp-Massey che consente di evitare la costosa operazione d'inversione della matrice, sfruttando la relazione lineare (del tipo LFSR) tra i valori di sindrome del codice e i coefficienti di un polinomio grazie al quale è possibile individuare le posizioni degli errori occorsi durante la trasmissione. Tale polinomio è chiamato per questo motivo polinomio di identificazione della collocazione degli errori. Dato che le operazioni in fase di decodifica, compresa la *error detection*, devono essere effettuate nel campo di Galois in cui giacciono le radici del polinomio generatore, è stata prestata particolare attenzione allo studio della struttura del codice BCH previsto dal DVB-S2 (completando le riduttive informazioni fornite dallo standard e concludendo che si tratta di un codice primitivo e shortened , ossia più corto rispetto alla sua lunghezza "naturale") e alla conseguente determinazione e costruzione di tabelle (del campo finito) ausiliarie per poter effettuare tutte le operazioni necessarie durante le operazioni di decodifica.

Infine, l'ultima parte della tesi si spinge a mostrare anche i risultati sinora raggiunti delle fasi di collaudo in laboratorio realizzate sull'intera sezione di trasmissione del sistema DVB-S2 sviluppata da TAS-I. Il progetto hardware è attualmente in corso di verifica mediante l'uso di una scheda di prototipazione Altera fondata su FPGA (*Field Programmable Gate Array*) EP180 Stratix II. Non si è ancora giunti

alla prova BER *end-to-end*, ma si sono effettuate verifiche delle prestazioni della sezione di modulazione ad elevata cardinalità. Nella descrizione si illustra brevemente l'architettura e i blocchi della intera sezione di trasmissione, vengono mostrati i risultati ottenuti attraverso un VSA (*Vector Signal Analyzer*) a banda larga, in grado di demodulare il segnale, interconnesso ad un oscilloscopio, entrambi prodotti Agilent. Le prestazioni della sezione di trasmissione sono state valutate attraverso l'analisi statistica degli errori di ampiezza e di fase (effettuata dal VSA). Tra elementi di innovazione associati alle elevate velocità di simbolo da raggiungersi, si è affrontato anche il problema della correzione della distorsione d'ampiezza del convertitore digitale-analogico (DAC, *Digital to Analog Converter*). Le misure effettuate mostrano infatti l'importanza del filtro di precompensazione quando le velocità di trasmissione dei dati sono più elevate.

Acknowledgements

Ringrazio il prof. Roberto Garello che mi ha dato la possibilità, mettendomi in contatto con l'ing. Domenico Giancristofaro, di svolgere l'attività di tesi a L'Aquila.

Desidero ringraziare la Thales Alenia Space Italia di L'Aquila e, segnatamente, il suo reparto di architetture e algoritmi, che mi ha messo a disposizione tutti gli strumenti e il materiale necessari allo svolgimento e completamento della mia attività di tesi.

Sono grato all'ing. Domenico Giancristofaro, che, nonostante i numerosi impegni, mi ha seguito con costanza e pazienza durante lo svolgimento della mia attività in TAS-I.

Infine, un ultimo ringraziamento agli ingegneri Cinzia Moca e Massimo Fonte, che mi hanno fornito degli ottimi spunti e consigli, utili allo svolgimento della mia attività e alla redazione della tesi.

Contents

Capitolo 1

Riassunto

1.1 Applicazioni e caratteristiche delle comunicazioni satellitari

Le comunicazioni satellitari rivestono un ruolo molto importante in applicazioni che potremmo definire di nicchia o altamente strategiche quali:

- La trasmissione e diffusione di dati multimediali su vaste aree geografiche a bassa densità di popolazione, dove per l'appunto è sconveniente investire in infrastrutture di reti di comunicazione.
- Le comunicazioni marittime (ad esempio Inmarsat) e i sistemi di radionavigazione, questi ultimi in particolare si stanno largamente diffondendo tra i prodotti consumer (si pensi ai noti navigatori satellitari TomTom per autoveicoli).
- Diffusione della TV via satellite: è spesso economicamente più conveniente coprire una vasta area geografica attraverso satelliti piuttosto che attraverso una grossa rete di stazioni locali.
- Telerilevamento e osservazione della Terra, settore altamente strategico sia per applicazioni militari sia per applicazioni civili.

In generale tutti questi tipi d'applicazione richiedono un alto *bit rate*. Tuttavia le risorse a bordo dei satelliti sono preziose e limitate: questo spiega l'alto interesse da parte delle industrie a promuovere la ricerca nel campo delle modulazioni e della codifica canale in modo da ottenere delle prestazioni il più possibile vicine a quelle limite.

Le comunicazioni satellitari per l'osservazione della Terra e le Telecomunicazioni sono quindi caratterizzate dall'esigenza di ottenere un alto throughput ed efficienze

in banda e potenza molto alte. Inoltre, la banda disponibile e i requisiti sulla velocità di segnalazione e sul rapporto segnale rumore possono variare sensibilmente a seconda dell'applicazione e della specifica missione spaziale. Per questo motivo un'unità modem sufficientemente flessibile da poter essere utilizzata per un numero elevato di missioni e applicazioni dovrebbe avere le seguenti peculiarità:

- Regolazione della velocità di trasmissione e della larghezza di banda
- Prestazioni molto vicine (a un 1 dB massimo) dal limite di Shannon, rispetto al tipo di modulazione e codifica utilizzate
- Utilizzo efficiente delle risorse hardware ed energetiche a bordo per l'acquisizione e l'elaborazione dei dati

Il sistema di modulazione e codifica adottato dal recente standard per le comunicazioni satellitari DVB-S2, così come MOHMS, un innovativo modem ad alto ordine di modulazione e alte prestazioni sviluppato da TAS-I e finanziato dall'European Space Agency (ESA), è così vicino alle prestazioni limite (Shannon) da poter rappresentare una solida soluzione tecnologica per molti anni a venire.

1.1.1 Osservazione della Terra e telerilevamento

Come già accennato, i satelliti ricoprono un ruolo chiave in applicazioni (civili e militari) per cui è richiesta la copertura di vaste aree di territorio a bassa densità di popolazione (per esempio, oceani, deserti, zone polari, foreste, etc.). La copertura di queste zone e l'alta capacità di acquisizione ed elaborazione di immagini fa sì che si possano fornire servizi quali, ad esempio, l'oceanografia, la meteorologia, la climatologia.

COSMO-Sky-Med è un eccellente esempio di sistema integrato di telecomunicazione in grado scambiare dati con altri sistemi (esterni) di osservazione della Terra rispettandone le modalità e i protocolli di comunicazione. COSMO-Sky-Med (COSMO è un acronimo inglese che sta per costellazione di quattro piccoli satelliti per l'osservazione del Mediterraneo) è composto di una costellazione di quattro satelliti, operanti in banda X e equipaggiati di radar ad apertura sintetica ad alta risoluzione, disposti in orbita bassa. Thales Alenia Space (TAS) è responsabile dello sviluppo e della costruzione dell'intero sistema sia per il segmento spaziale sia per il segmento di terra.

Le prestazioni di COSMO-Sky-Med sono caratterizzate da tre aspetti: il basso tempo di rivisita, il basso tempo di risposta del sistema e le alte prestazioni fornite dal radar ad apertura sintetica a modalità operative multiple. Le alte prestazioni nella fase di acquisizione delle immagini sono garantite dal versatile radar, ad apertura sintetica, in grado di acquisire immagini in differenti modalità operative,

generando quindi immagini a vari livelli di risoluzione e dimensione in modo da coprire un ampio spettro di esigenze applicative.

I radar ad apertura sintetica (in inglese, Syntetic Aperture Radar) sono largamente usati nelle missioni il cui scopo è la raccolta di grossi volumi di immagini ad alta risoluzione. In questo campo Thales Alenia Spazio ricopre un ruolo importante nello sviluppo dell' X-SAR, radar ad apertura sintetica operante in banda X. L'X-SAR è in grado di misurare quasi ogni regione del Globo in qualsiasi condizione meteorologica e di luce, nonché di penetrare folta vegetazione, ghiaccio e sabbia in modo da poter fornire agli scienziati informazioni dettagliate sul clima e processi geologici, così come sui cicli idrologici e le correnti oceaniche.

Per quanto riguarda le applicazioni di COSMO-Sky-Med, la costellazione e la versatilità del SAR, a modalità operative multiple, consente di ottenere un'ampia varietà di immagini di differenti dimensioni e risoluzioni, con differenti livelli di accuratezza e di elaborazione, adatte pertanto ad applicazioni come il controllo e la sorveglianza di vaste zone di territorio, nonché al rilevamento di fenomeni critici. Attraverso le immagini ricavate per interferometria possono essere ottenuti modelli tridimensionali e planimetrici (*Digital Elevation Models*) di città, costruzioni, strade, e/o rilievi topografici ad alta risoluzione.

Le applicazioni bidimensionali (2D) sono volte specialmente all'effettuazione di rilievi cartografici. Altre applicazioni possibili sono:

- Agricoltura
- Classificazione di zone geografiche
- Rilevamento e localizzazione di risorse geologiche come olio, gas e minerali
- Accurate mappe di zone urbane per stabilire l'evoluzione demografica e per la gestione della stessa (ad esempio, per mezzo dell'identificazione delle zone ad alta/bassa densità di popolazione), così come mappe di zone industriali insieme all'identificazione di costruzioni di interesse, zone aeroportuali, etc.
- Rilevamento e localizzazioni di infrastrutture di rete (gas, telefoni, strade) per la compilazione della banca dati GIS (Geographic Information System).

Altri campi di applicazione riguardano l'idrologia a il rilevamento del profilo costiero. Le applicazioni tridimensionali (3D) sono varie e comprendono:

- *Aiuto decisionale.* Modelli tridimensionali forniscono un aiuto ai cittadini e ai sindaci nella comprensione e conoscenza del territorio e nella eventuale costruzione di una banca dati per poter effettuare delle simulazioni sulla base dei modelli acquisiti. Per esempio, tramite questo sistema si potrebbe valutare, per mezzo di simulazioni, l'impatto ambientale di una nuova infrastruttura sulla città.

- *Gestione delle componenti di rischio industriale e ambientale*: simulazione, prevenzione, assistenza e assistenza *post* incidente. L'uso di modelli tridimensionali e planimetrici combinati a immagini ad alta risoluzione consente la simulazione di fenomeni come allagamenti, incendi, terremoti, inquinamento atmosferico, etc.

1.1.2 La seconda generazione dello standard DVB-S

Sviluppato sul successo del DVB-S, il DVB-S2 è lo standard di seconda generazione per applicazioni satellitari a banda larga. Esso beneficia dei recenti progressi tecnologici ottenuti nello scorso decennio. Questo standard è stato progettato per svariate applicazioni a banda larga:

- Servizi diffusivi di TV a definizione standard (SDTV, Standard Definiton TeleVision) e ad alta definizione (HDTV, High Definition TeleVision)
- Applicazioni interattive per l'utenza domestica e professionale, compreso l'accesso ad Internet
- Servizi professionali di contribuzione TV ed SNG (Satellite News Gathering)
- Distribuzione di segnali TV a trasmettitori digitali terrestri VHF/UHF
- Distribuzione dati e di siti Internet (Internet trunking).

Sono tre i concetti chiave in base a cui lo standard DVB-S2 è stato definito: maggiore capacità trasmissiva rispetto ai sistemi di prima generazione e in particolare al DVB-S, totale flessibilità, ragionevole complessità del ricevitore. Per ottenere il bilanciamento tra prestazioni e complessità, il DVB-S2 si avvale dei più recenti sviluppi nella codifica di canale e nella modulazione.

L'adozione nel DVB-S2 di queste innovative tecniche di codifica e modulazione garantisce un aumento di capacità dell'ordine del 30% rispetto al DVB-S nelle stesse condizioni di trasmissione, in modalità CCM (*Constant Coding & Modulation*, letteralmente Modulazione e Codifica Costanti), ossia con parametri di trasmissione fissi.

La codifica di canale del DVB-S2 è basata sui codici LDPC (*Low Density Parity Check*), una famiglia di codici a blocco molto semplici, con una struttura algebrica molto limitata, scoperti da Robert Gallager nel 1962. Questi codici hanno degli algoritmi di decodifica facilmente eseguibili in maniera parallela, consistenti in operazioni elementari come addizioni, confronti e letture da memoria. Inoltre il grado di parallelismo di questi algoritmi di codifica è facilmente modulabile, rendendo così semplice trovare il giusto compromesso tra complessità e throughput.

Le caratteristiche chiave che consentono di raggiungere prestazioni a soli 0,6-1,2 dB dal limite di Shannon sono:

- la grande lunghezza del blocco di codifica LDPC (64800 bit per blocchi cosiddetti normali e 16200 bit per blocchi corti);
- l'elevato numero di iterazioni in decodifica (circa 50 interazioni SISO); deve essere sottolineato che la struttura di codifica mostra periodicità utilizzabili per realizzare decodificatori con alto parallelismo;
- la concatenazione con un codice esterno BCH (Bose-Chaudhuri-Hocquenghem) (senza nessun interlacciamento), definito dai progettisti come un "margine di sicurezza a basso costo contro eventuali errori residui non prevedibili ad elevati rapporti C/N" (*error floor*).

Nelle applicazioni punto-punto, come l'IP Unicast, il guadagno del DVB-S2 rispetto al DVB-S può essere ancora maggiore. La funzionalità ACM (*Adaptive Coding & Modulation*, letteralmente modulazione e codifica adattative) permette infatti di variare lo schema di modulazione ed i livelli di protezione dagli errori ad ogni nuovo blocco elementare di codifica, ottimizzando il sistema di trasmissione alle condizioni di ricezione d'utente. Per informare il trasmettitore delle condizioni di ricezione del singolo utente, il sistema deve operare 'ad anello chiuso', utilizzando un canale di ritorno via telefono o satellite.

Il DVB-S2 è così flessibile da adattarsi a tutti i tipi di transponder satellitari esistenti, grazie ad un'ampia varietà di efficienze spettrali e rapporti segnale-rumore C/N richiesti. Inoltre, è progettato per trattare una grande varietà di formati audio-video e di dati, dall'MPEG-2 attualmente utilizzato negli standard DVB, a quelli che il progetto DVB sta attualmente definendo per le applicazioni future (H264 e VC9). Il sistema DVB-S2 si adatta a qualunque formato di flusso di dati in ingresso, compresi flussi digitali MPEG Transport Stream (TS), singoli o multipli, IP e ATM. Questo fa sì che, anche se in futuro verranno definiti altri formati, essi potranno essere impiegati senza bisogno di modificare il sistema.

Per quanto concerne i possibili scenari applicativi, il DVB-S2 è stato ottimizzato per i seguenti servizi a banda larga:

Servizi diffusivi (Broadcast Services) Si tratta della diffusione del segnale (in formato MPEG) radiotelevisivo (HDTV, SDTV) non solo agli utenti dotati di decoder e ricevitore digitale (Integrated Receiver Decoder), ma anche a sistemi di raccolta del segnale (Satellite Master Antenna TeleVision) e a stazioni di inoltro del segnale via cavo. Da questo punto di vista il DVB-S2 può essere visto come il diretto successore del DVB-S, lo standard corrente. La tecnica di modulazione e codifica variabile ad anello aperto (Variable Coding and Modulation, l'equivalente dell'ACM senza l'utilizzo del canale di ritorno) può essere usata in modalità di trasporto multiplo per ottenere un livello di protezione contro gli errori a seconda dei differenti servizi (TV, HDTV, audio, dati multimediali).

Servizi interattivi (Interactive Services) Il DVB-S2 è in grado di fornire servizi interattivi, tra cui l'accesso a internet, ai possessori di decoder digitali (consumer, ossia prodotti ad alta diffusione) e di personal computer. Dato che lo standard lascia piena libertà sul canale di ritorno, l'interattività potrebbe essere stabilita sia attraverso canali terrestri (utilizzando evidentemente le linee telefoniche) sia attraverso l'uso di satelliti. I dati possono essere trasportati in formato MPEG oppure in formato generico, in modalità CCM (Constant Coding and Modulation), ossia a modulazione e codifica costanti, oppure in ACM. In quest'ultimo caso ogni stazione ricevente controlla il livello di protezione dei dati ad essa indirizzati.

1.2 La sezione di trasmissione del DVB-S2: concetti e architettura

1.2.1 Schemi di modulazione e codifica

In ogni sistema di comunicazione la scelta ed il progetto degli schemi di modulazione e codifica ha come finalità quella di trasmettere i dati in maniera sufficientemente affidabile per il tipo di applicazione richiesta, facendo al tempo stesso un uso efficiente delle risorse disponibili (banda, potenza, ed, in particolare nelle comunicazioni via satellite, peso ed ingombro delle apparecchiature). Le quantità e i requisiti che vengono tipicamente valutati nella scelta di un particolare schema di modulazione e codifica sono i seguenti

- La probabilità d'errore (BER, *Bit Error Rate*), che dà un'utile indicazione su quanto sia affidabile la trasmissione di messaggi.
- L'efficienza spettrale, che misura l'efficienza di utilizzo della banda disponibile
- Il rapporto segnale rumore (SNR, *Signal to Noise Ratio*) necessario per raggiungere una desiderata qualità del servizio (QoS, *Quality of Service*) . Questo parametro indica quanto lo schema di modulazione e codifica scelto sfrutti efficientemente la potenza disponibile.
- La complessità realizzativa, che rappresenta una misura del 'costo' delle apparecchiature.

Nel 1948, Claude E. Shannon definisce e quantifica l'informazione, riuscendo inoltre a dimostrare che, per una qualsivoglia velocità di trasmissione minore o uguale a un cardinale parametro chiamato capacità di canale, esiste uno schema di codifica in grado di ottenere una probabilità d'errore arbitrariamente piccola, garantendo così una trasmissione completamente affidabile dei dati. Purtroppo egli nel suo teorema

non dà alcuna indicazione su come poter trovare questi schemi di codifica. Shannon dimostra tuttavia che codici lunghi scelti casualmente assicurano una bassa probabilità d'errore media. Purtroppo una diretta realizzazione dei codici suddetti porta ad avere una complessità di decodifica molto alta. Dal 1948, quindi, l'impegno degli ingegneri delle telecomunicazioni si concentra sullo sviluppo di schemi realizzabili di modulazione e codifica, che si avvicinino alle prestazioni limite. Nonostante l'iniziale pessimismo, il problema viene risolto almeno per il caso di maggior interesse e importanza, il canale lineare gaussiano bianco (AWGN, *Additive White Gaussian Noise*).

Nel 1980, poi, la migliore comprensione del significato del teorema di Shannon porta a una revisione del paradigma utilizzato sino ad allora nel campo delle modulazioni e della codifica. Queste due discipline, che prima di questo momento si sviluppavano indipendentemente, incominciano a diventare rigidamente collegate. Parallelamente, l'esigenza di ottenere alte efficienze spettrali fa aumentare la cardinalità delle modulazioni (il numero delle forma d'onda impiegate in ogni simbolo); contestualmente, vengono ideati codici a correzione d'errore ancora più efficaci. In tale contesto il lavoro di Ungerboek sancisce l'arrivo dei codici TCM (*Trellis Coded Modulation*), rendendo chiari i vantaggi del trattare modulazione e codifica come una entità singola.

Inoltre, vari schemi pubblicati in letteratura dimostrano che Turbo Codici (una tecnica di codifica introdotta nel 1993 da Claude Berrou) e TCM possono essere utilizzati con profitto congiuntamente. Tuttavia i Turbo Codici dovrebbero essere progettati ad hoc per ogni singola modulazione, una soluzione impraticabile per sistemi che adottano più costellazioni. La soluzione di questo problema prende il nome di *Bit Interleaved Coding Modulation* (BICM), tecnica grazie alla quale è possibile ottenere dei risultati eccellenti, impiegando una singola soluzione di codifica per tutte le modulazioni previste. Questa grande flessibilità rende le BICM commercialmente appetibili per sistemi comunicazione satellitari a banda larga, di tipo adattativo (*Adaptive Coding Modulation*, ACM), che necessitano di un ampio ventaglio di efficienze spettrali. Appartengono alle soluzioni ACM anche lo standard ETSI DVB-S2 e la soluzione MHOMS in via di standardizzazione nel CCSDS (Consultive Committee for Space Data Systems).

Il recente standard DVB-S2, successivo al già consolidato DVB-S, utilizza gran parte delle più recenti e innovative tecniche nel campo della codifica di canale e delle modulazioni congiuntamente a modulazioni ad alta cardinalità, come la 16APSK e la 32APSK. Più precisamente il DVB-S2 fornisce quattro schemi di modulazione

- QPSK e 8-PSK, che sono tipicamente impiegati nelle applicazioni di radiodiffusione, poiché hanno un inviluppo costante e quindi possono essere utilizzati agevolmente su trasponder satellitari vicini al regime di saturazione dei tipici amplificatori TWTA, *Travelling Wave Tube Amplifier*.

- 16-APSK a 32-APSK, che possono anch'essi essere utilizzati in servizi diffusivi, tenendo presente che richiedono un maggior rapporto C/N e una maggiore complessità, poiché richiedono l'adozione di sofisticate tecniche di predistorsione delle costellazioni.

La tecnica di modulazione e codifica adattativa, *Adaptive Coding and Modulation* (ACM), definisce una strategia innovativa per impiegare le risorse al meglio e aumentare la capacità di trasmissione dati: infatti, l'impiego di una strategia flessibile che ottimizza l'impiego delle risorse disponibili in quel determinato momento ne permette un più efficace sfruttamento; ciò in base alle condizioni di attenuazione del radiocollegamento.

Il concetto di base è utilizzare un criterio di assegnazione delle risorse, variabile nel tempo, che tenga conto del requisito istantaneo di potenza degli utenti; infatti, in una tratta radio l'attenuazione, ad esempio, è variabile nel tempo; quindi conviene inviare un segnale più potente in condizioni di attenuazione molto elevata mentre è preferibile ridurre la potenza del segnale trasmesso, quando l'attenuazione del canale è bassa; in entrambi i casi, il rapporto segnale rumore al ricevitore dovrà essere maggiore o uguale a una certa soglia.

In definitiva, le tecniche ACM sono indicate per i canali radiomobili, che hanno una risposta tempo variante: una trasmissione adattativa permette di sfruttare in maniera più efficiente la capacità del canale. La tecnica ACM può essere alternativa o combinata alle seguenti tecniche di mitigazione degli affievolimenti che si verificano nel canale di trasmissione.

- **Controllo dinamico di potenza.** Attraverso il controllo di potenza, il livello di segnale trasmesso varia in accordo con le fluttuazioni del canale. Questa strategia incrementa il livello massimo di potenza e nel caso di un ambiente multiutente il livello di interferenza co-canale, che può ridurre la capacità di canale se la coordinazione tra gli utenti non è permessa.

- **Dimensione della costellazione adattativa.** La modulazione adattativa gioca un ruolo importante perché consente l'incremento dell'efficienza di trasmissione dei dati. Il concetto consiste nel trasmettere con il tasso di informazione più elevato possibile per il livello di qualità del servizio assegnato, specificato in termini di probabilità di errore. La modulazione adattativa si ottiene usando una gerarchia di costellazioni differenti ordinate in dimensione crescente.

- **Rapporto di codifica adattativo.** Lo schema di codifica viene cambiato in risposta allo stato del canale di trasmissione, selezionando il miglior compromesso sul rapporto di codifica. A questo scopo sono particolarmente indicati i codici che consentono una codifica e decodifica adattativa senza modificare la

struttura di base del codec, come quelli basati su un approccio pragmatico o Interleaved bit modulation.

- **Livello di potenza e dimensione della costellazione adattativa.** La combinazione può essere impiegata su un canale diviso o condiviso da più utenti, permettendo un incremento del throughput significativo rispetto all'assenza del controllo dinamico di potenza.
- **Dimensione della costellazione e rapporto di codifica adattativi.** La combinazione permette al sistema di scegliere la modulazione insieme al rapporto di codifica, raggiungendo in modo ottimale le prestazioni richieste in termini di probabilità d'errore.
- **Livello di potenza e tasso di codifica adattativi.** La combinazione permette di massimizzare l'efficienza spettrale garantendo un livello di potenza medio e le prestazioni richieste.
- **Livello di potenza, dimensione della costellazione e tasso di codifica adattativi.** Il livello di potenza, la dimensione della costellazione e il tasso di codifica possono essere adattati simultaneamente.

Se il valore di BER richiesto non è ottenibile tramite nessuna delle combinazioni, allora il sistema non trasmette dati.

1.2.2 Descrizione della sezione di trasmissione

Il sistema DVB-S2 è stato progettato in base a due livelli di trama del segnale:

- il primo, a livello fisico (PL, *Physical Layer*), che trasporta pochi bit di segnalazione molto protetti;
- il secondo, a livello di banda base (BB, *Base Band*), che trasporta molti bit di segnalazione, per consentire la massima flessibilità di adattamento del segnale di ingresso.

Il primo livello di trama è stato progettato in modo tale da consentire di rivelare la modulazione e i parametri di codifica prima della demodulazione e della decodifica FEC (*Forward Error Correction*) e garantire la possibilità di sincronizzare il ricevitore (recupero di portante e fase, sincronizzazione di trama) in condizioni di C/N molto critiche, dettate dalle alte prestazioni del FEC.

Il livello fisico del DVB-S2 è composto da sequenze regolari e periodiche di dati, denominate PLFRAME, costituenti la trama di livello fisico: all'interno di Uno stesso PLFRAME, lo schema di modulazione e codifica è omogeneo, ma può variare in modalità VCM (*Variable Coding & Modulation*) tra PLFRAME adiacenti.

Indipendentemente dall'applicazione (CCM o VCM), ogni PLFRAME è composto da:

- un carico utile FECFRAME di 64800 bit (FECFRAME normale) o 16200 bit (FECFRAME corto), corrispondente a un blocco codificato LDPC/BCH, generato codificando i bit d'utente secondo lo schema FEC scelto;

- l'intestazione del PLFRAME denominata PLHEADER, contenente informazioni per la sincronizzazione e la decodifica: tipo di modulazione e tasso di codifica FEC, lunghezza del FECFRAME, presenza/assenza di simboli pilota per facilitare la sincronizzazione.

L'intestazione del PLFRAME è composta sempre da 90 simboli (che usano una modulazione binaria Π/2 BPSK) e il carico utile da un numero intero multiplo di 90 simboli (ad esclusione dei simboli pilota).

Poiché l'intestazione del PLFRAME è la prima entità ad essere decodificata dal ricevitore, non può essere protetta dal potente schema LDPC/BCH. D'altra parte esso deve poter essere ricevuto correttamente anche nelle peggiori condizioni di collegamento; si è pertanto ridotto al minimo (7) il numero di bit di segnalazione, per diminuire la perdita di efficienza globale, e per ridurre la complessità della decodifica sono stati protetti con un codice a blocco specifico con tasso di codifica molto basso 7/64, adatto per decodifica a correlazione con *soft-decision*. Nel caso peggiore, assumendo un FECFRAME di 64800 bit, l'efficienza del PLFRAME è 99,3% (in assenza di simboli pilota).

La trama in banda base permette invece una segnalazione più completa della configurazione trasmissiva, con indicazione della molteplicità dei flussi d'ingresso (singolo o multiplo), del tipo (generico GS, dall'inglese *Generic Stream* o TS, *Transport Stream*), e della modalità di trama, CCM o ACM. Grazie alla protezione del codice FEC LDPC/BCH e alla lunghezza dei blocchi di codifica, l'intestazione del blocco elementare della struttura di banda base, denominato BBFRAME, può contenere molti bit di segnalazione (80), senza perdere efficienza trasmissiva e neppure robustezza contro il rumore.

L'intestazione BB trasporta quindi altre importanti informazioni di segnalazione come: etichetta dei flussi all'ingresso del modulatore, descrizione della posizione e delle caratteristiche dei pacchetti d'utente, indicazione della presenza di bit di riempimento (padding bits) nel BBFRAME trasmesso, segnalazione della messa in funzione di specifici strumenti (funzione di cancellazione dei pacchetti nulli (*null packets*), funzione di sincronizzazione del flusso di ingresso, segnalazione del coefficiente di *roll-off* adottato.

Il sistema DVB-S2 è composto da una sequenza di blocchi funzionali.

Il blocco, identificato come Adattatore di modo e di flusso, svolge funzioni legate all'applicazione. Esso fornisce l'interfaccia per il flusso di ingresso, strumenti opzionali richiesti per l'ACM (ad esempio per la sincronizzazione e la cancellazione dei pacchetti nulli nel caso di flussi di ingresso di tipo TS) e inserisce la codifica CRC (*Cyclic Redundancy Check*) per permettere al ricevitore di rivelare la presenza di errori nel flusso ricevuto. Oltre a ciò, nel caso di ingressi multipli, esso unisce i flussi di ingresso (*Merger*) per poi suddividerli (*Slicer*) in blocchi del codice FEC. Questi ultimi sono composti da bit presi da una sola porta di ingresso da trasmettere in modo omogeneo (stessa modulazione e codice FEC).

Si inserisce poi l'intestazione di banda base (80 bit) davanti al Campo Dati per informare il ricevitore del formato del flusso di ingresso e del tipo di 'adattamento' utilizzato. Nel caso i dati utente disponibili per la trasmissione non siano sufficienti a riempire completamente il BBFRAME, si provvederà a completarlo con bit di riempimento. In ultimo, nel blocco denominato *"Stream Adapter"* il BBFRAME viene moltiplicato per una sequenza pseudocasuale (*Scrambler*), che uniformemente distribuisce gli zeri e gli uno del BBFRAME, evitando la presenza di sequenze critiche per il codice FEC.

Il blocco Codifica FEC effettua la codifica concatenata del codice esterno BCH e del codice interno LDPC. I rapporti di codifica del codice LDPC interno sono 1/4, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5, 5/6, 8/9, 9/10, da scegliersi, congiuntamente allo schema di modulazione, in base ai requisiti del sistema. I rapporti 1/4, 1/3 e 2/5 sono stati introdotti per operare in combinazione con lo schema di modulazione QPSK, per collegamenti di bassa qualità, dove il livello del segnale è al di sotto del livello di rumore. Le simulazioni al computer hanno dimostrato la superiorità di tali modalità rispetto alla modulazione BPSK combinata con velocità di codifica 1/2, 2/3 e 4/5. A seconda dell'area di applicazione i blocchi di codice FEC (FECFRAME), possono avere una lunghezza di 64800 o 16200 bit. L'introduzione di due possibili valori è stata dettata da due opposte necessità. Le prestazioni in funzione del rapporto C/N migliorano al crescere della lunghezza dei blocchi di codifica, ma aumenta, anche molto, il ritardo globale della catena trasmissiva. Quindi, per applicazioni non critiche per i ritardi (come ad esempio la diffusione di programmi), sono preferibili i blocchi lunghi, mentre per le applicazioni interattive un blocco più corto può essere più efficiente, in quanto un pacchetto di informazione corto viene immediatamente messo in onda dalla stazione trasmittente. La modulazione e il codice FEC sono costanti all'interno del FECFRAME, e possono cambiare in differenti FECFRAME nelle modalità VCM e ACM. Oltre a ciò il segnale trasmesso può contenere FECFRAME corti e normali. Per le modulazioni 8PSK, 16APSK e 32APSK ai bit codificati FEC si applica un interallacciatore di bit, per separare i bit assegnati allo stesso punto della costellazione in trasmissione.

Il blocco di Mapping associa i bit alla costellazione: QPSK, 8PSK, 16APSK o 32APSK a seconda dell'applicazione. Tipicamente, per applicazioni broadcast vengono proposte le costellazioni QPSK e 8PSK, poiché sono di fatto modulazioni ad inviluppo costante e possono essere usate su transponder da satellite non lineari portati vicino alla saturazione. Le modalità 16APSK e 32APSK invece sono principalmente orientate ad applicazioni professionali; possono anche essere impiegate per il *broadcasting*, ma richiedono la disponibilità di un più elevato livello di C/N al ricevitore e l'adozione di avanzati metodi di pre-distorsione nella stazione di up-link per attenuare gli effetti di nonlinearità del transponder. Sebbene non permettano efficienze di potenza analoghe agli schemi ad inviluppo costante, offrono però maggiore capacità trasmissiva.

Il blocco di Generazione della trama PL, sincrono con i FECFRAME, gestisce l'inserzione dell'intestazione di livello fisico e dei simboli pilota opzionali (2,4 % di perdita di capacità), di PLFRAME fittizi (*Dummy Frame*) in assenza di dati utili pronti per la trasmissione, e la moltiplicazione per una sequenza pseudocasuale (*Scrambler*) per la dispersione dell'energia.

Il filtraggio in banda base e la modulazione in quadratura si applicano per modellare lo spettro del segnale e per generare il segnale RF (*Radio Frequency*). Il filtro usato in trasmissione è la radice quadrata del filtro a coseno rialzato con tre possibili coefficienti di *roll-off* a: 0,35 per continuità con il DVB-S, 0,25 e 0,20 per i casi in cui vi siano maggiori limitazioni di banda.

1.2.3 La codifica di canale

La FEC del DVB-S2 si basa sulla concatenazione di due codici a blocco: BCH e LDPC. Questa concatenazione è stata dettata dall'esigenza di limitare i tipici cali di pendenza della curva di probabilità d'errore ad alti rapporti segnale rumore. Tale fenomeno porta il nome di *error floor* e causa un appiattimento delle curve di BER (*Bit Error Rate*) dopo le regioni ripide (le regioni a "cascata", dall'inglese *waterfall*, tipiche della decodifica iterativa) a più bassi rapporti segnale rumore.

I codici BCH prendono il loro nome da Bose, Ray-Chaudhury e Hocquenghem, che nel 1959 e '60 indicarono un modo di progettare codici a alfabeto binario con una specifica distanza minima. I codici BCH appartengono alla famiglia dei codici ciclici e quindi possono essere specificati da un generatore polinomiale.

Il generatori polinomiali dei codici ciclici vengono tipicamente ricavati dal prodotto di alcuni polinomi, chiamati minimi, derivanti a loro volta dalla fattorizzazione di $x^n - 1$ su un arbitrario campo finito. Ulteriori dettagli su questo genere di argomenti sono reperibili nelle due appendici.

Il metodo di progetto di un codice BCH in grado di correggere t errori utilizza estensivamente l'algebra dei campi finiti e la teoria dei codici ciclici, a cui sono state dedicate due appendici (l'appendice A sui campi di Galois e l'appendice B sui

codici ciclici). Dalla tabella 3.4 dei polinomi minimi fornita nello standard DVB-S2, in accordo con la procedura di progetto dei codici BCH, è possibile ottenere un generatore polinomiale del codice per ogni capacità correttiva prevista dalla codifica BCH esterna (si osservino le espressioni (5.2), (5.3), (5.4) nel capitolo dedicato all'architettura hardware del codificatore BCH). Dalla teoria dei codici ciclici risulta che ogni parola di codice - risultante dal calcolo del prodotto o, come nel DVB-S2, del resto della divisione del messaggio per il generatore polinomiale del codice - deve essere un multiplo del generatore polinomiale. Infine, poiché tra i polinomi minimi riportati nella tabella 3.4 è presente il polinomio primitivo di $GF(2^{16})$, risulta evidente che il codice BCH usato nel DVB-S2 deve essere primitivo. Da questo consegue che la lunghezza delle parole di codice BCH, avendo le radici del polinomio generatore giacenti in $GF(2^{16})$ ed essendo primitivo, deve essere $2^{16} - 1$. Questo implica che il codice sia pure *shortened*. Un codice è chiamato *shortened* quando la 'naturale' lunghezza delle parole di codice viene "accorciata". Questo troncamento non ha, in linea di principio, ripercussioni negative sulle proprietà di distanza minima del codice e su alcuni algoritmi di codifica. Vale la pena ancora di osservare che in fase di codifica è sufficiente avere la sola conoscenza del polinomio generatore del codice, mentre in fase di decodifica occorre conoscere e determinare le sue radici.

I codici LDPC furono scoperti da Gallager nel 1961, tuttavia ad essi è stata data importanza dopo l'invenzione dei Turbo codici e del principio della decodifica iterativa da parte di Berrou, Glavieux e Thimajshima nel 1993. Ciò ha permesso la decodifica di parole con blocchi di elevata lunghezza con complessità ragionevole, senza ricorrere ad algoritmi di decodifica a massima verosimiglianza.

Nel progetto dei codici a blocchi lineari con blocchi di breve lunghezza, il problema è quello di realizzare strutture la cui distanza minima è la più elevata possibile in modo tale da ottenere una elevatà capacità in termini di rilevazione e di correzione degli errori. L'inconveniente principale dell'approccio classico è che nell' ipotesi di parole con blocchi di elevata lunghezza, trovare dei codici con elevata distanza minima è estremamente complicato.

Nella codifica iterativa, i codici vengono costruiti in modo tale che le relazioni con i bit di ridondanza siano localmente semplici, mentre tali descrizioni locali del codice sono interconnesse tramite relazioni piuttosto complesse, ad esempio di tipo casuali. La decodifica iterativa calcola continuamente le semplici descrizioni elaborando iterativamente i risultati ottenuti, che vengono passati sotto forma di messaggi attraverso le complesse interconnessioni. Solitamente si adotta un grafo (*Tanner Graph*) per descrivere il processo di decodifica iterativa, in cui le equazioni di controllo di parità rappresentano i nodi che elaborano le informazioni e le interconnessioni rappresentano i rami che scambiano le informazioni.

Il successo della decodifica iterativa, rispetto alla decodifica a massima verosimiglianza che caratterizza il progetto di codici a blocchi lineari, si intuisce dall'osservazione empirica secondo la quale, nel caso di parole di codice di elevata lunghezza

(almeno 1000 bit), per la nuova generazione di codici *Low Density Parity Check* (LDPC), la rappresentazione grafica è particolarmente sparsa e sono caratterizzati da bassa complessità ed eccellenti prestazioni in termini di probabilità di errore.

I codici a blocchi lineari generici hanno approssimativamente lo stesso numero di "0" e di "1" nella matrice di controllo di parità che non è sparsa; invece, nei codici LDPC il numero di "1" è veramente piccolo rispetto al numero degli "0" e la matrice è a bassa densità di controlli di parità, cioè il grafo associato al codice è a bassa densità di rami. L'osservazione fatta è fondamentale perché la complessità degli algoritmi di decodifica iterativi dipende direttamente dalla densità dei rami del grafo, di conseguenza si è interessati a progettare di codici in cui la specifica è quella di mantenere il più possibile bassa tale densità.

Gallager ha dimostrato negli anni '60 che una volta definite le prestazioni dei codici LDPC in termini di guadagno di codifica, cioè definita la distanza d dalla capacità di canale per una data probabilità di errore, è possibile stabilire il numero di controlli di parità necessari. Naturalmente, quanto più le prestazioni tendono al limite fisico di Shannon sulla capacità di canale, tanto più è richiesta una loro crescita. Inoltre l'andamento della complessità di decodifica è in netto contrasto con gli schemi di decodifica a massima verosimiglianza che soffrono di una crescita esponenziale della complessità quando si trasmette prossimi alla capacità di canale.

L'osservazione fatta spiega la ragione per la quale la decodifica iterativa, rispetto alla decodifica a massima verosimiglianza, è più promettente per una trasmissione affidabile delle informazioni, tendente al limite fisico di Shannon.

L'algoritmo di decodifica LDPC, valido nel caso in cui l'alfabeto dei messaggi è continuo con l'uscita del canale discreta, è denominato *Belief Propagation*. In generale, gli algoritmi di decodifica iterativi possono operare sia nel dominio delle probabilità, sia nel dominio logaritmico, che è più stabile numericamente. Nel dominio logaritmico, i messaggi passati in entrambe le direzioni attraverso i rami del grafo associato al codice, sono i logaritmi dei rapporti di verosimiglianza dei bit trasmessi (LLRs), che vengono calcolati a partire da quelli inizialmente forniti dal demodulatore al decodificatore.

La proprietà fondamentale dei decodificatori LDPC, che sono di tipo *Soft-Input Soft-Output*, consiste nel garantire che ad ogni iterazione venga scambiata attraverso i rami del grafo solo l'informazione estrinseca dei messaggi passati tra i *variable nodes* e i *check nodes*. Il requisito essenziale del concetto di informazione estrinseca è che il messaggio in uscita dal generico *check node* nella direzione del generico *variable node* non dipende dal messaggio precedentemente inviato nel verso opposto dallo stesso *variable node*. Analogamente, il messaggio in uscita dal generico *variable node* nella direzione del *check node* non dipende dal messaggio precedentemente inviato nel verso opposto dallo stesso *check node*.

I *check nodes* effettuano il controllo delle equazioni di parità quindi presentano

una struttura hardware molto più complessa di quella dei variable nodes. Tale struttura di controllo può essere semplificata nell'ipotesi in cui i messaggi da elaborare rappresentino una sequenza di variabili aleatorie linearmente indipendenti. Nella pratica i messaggi non sono linearmente indipendenti, a causa della presenza di cicli nel grafo associato al codice LDPC.

La seguente trattazione permette di chiarire l'algoritmo di codifica LDPC, integrando la descrizione fornita dalla Hughes Network System (HNS) che è stata utilizzata per la definizione dello standard DVD-S2.

I parametri del codice HNS-LDPC dipendono dal rapporto di codifica e dalla lunghezza dei blocchi delle parole di codice, che può essere di tipo *normal* FECFRAME ($n = 64800$) oppure di tipo *short* FECFRAME ($n = 16200$).

La matrice di controllo di parità ha una struttura bipartita. Si suppone che la codifica di canale permetta di conservare la sequenza informativa emessa dalla sorgente alle prime posizione della parola di codice. Tale affermazione equivale a dire che il codice HNS-LDPC è un codice sistematico.

La generica parola di codice è suddivisa nella parte informativa e nella parte dedicata ai bit di ridondanza. L'operazione di codifica consiste nella determinazione dei bit di parità che soddisfano un sistema di equazioni lineare omogeneo. Poiché, come appena detto, tale sistema è omogeneo e sia la matrice e sia il vettore della parola di codice sono bipartiti, la risoluzione numerica del problema può essere efficientemente ottenuta:

- determinando un vettore ausiliario (il numero di operazioni di XOR necessarie a questo scopo risulta essere una costante associata ai rapporti di codifica previsti dal DVB-S2)
- determinando i bit di parità, attraverso un metodo chiamato *back substitution*. Si tratta di un processo di inversione di matrice, che però, presentando elevate proprietà di regolarità, è possibile effettuare a basso 'costo' via hardware.

Le elevate proprietà di regolarità delle matrici consente inoltre di ottenere un notevole risparmio di memoria. Per concludere, la complessità di codifica cresce linearmente con la dimensione dei blocchi della parola di codice.

1.2.4 Cenni sulle prestazioni raggiunte

Il DVB-S2 permette di selezionare lo schema di modulazione e il tasso di codifica a seconda dei requisiti del servizio e delle caratteristiche del transponder da satellite impiegato. L'efficienza spettrale va da $0,5$, usando la modulazione QPSK $1/4$, a $4,5 bit/s/Hz$, usando la configurazione 32-APSK $9/10$, e il rapporto segnale rumore da $-2,4$ dB a 16 dB (assumendo canale AWGN e demodulazione ideale). I risultati sono stati ottenuti attraverso simulazioni al calcolatore valutanti le prestazioni dei

sistemi DVBS2 e DVB-S/DVB-DSNG ad un tasso d'errore sul pacchetto (PER, Packet Error Rate) TS di 10^{-7}, corrispondente circa a un pacchetto errato per ora di trasmissione in un servizio video a 5 Mbit/s. Su canale ideale affetto esclusivamente da rumore additivo Gaussiano bianco AWGN (Additive White Gaussian Noise), il risultato è un aumento della capacità trasmissiva dell'ordine del 20-35% rispetto al DVB-S e DVB-DSNG nelle stesse condizioni di trasmissione, o una ricezione di 2-2,5 dB più robusta per la stessa efficienza spettrale.

Il sistema DVB-S2 può essere usato nelle configurazioni 'singola portante per transponder' o 'multiportante per transponder' (Multiplazione a divisione di frequenza FDM, *Frequency Division Multiplexing*).

Nella configurazioni a singola portante per transponder, la velocità di trasmissione R_s (*symbol rate*) può essere adattata alla larghezza di banda BW del transponder (a -3dB), per ottenere la massima capacità trasmissiva compatibile con un degradamento accettabile del segnale dovuto alle limitazioni della larghezza di banda del transponder. Nella configurazione multi-portante FDM, R_s deve essere adattato all'intervallo di frequenza BS assegnato al servizio dal piano delle frequenze, per ottimizzare la capacità trasmissiva e contemporaneamente mantenere ad un livello accettabile le reciproche interferenze tra le portanti adiacenti.

1.3 Algoritmi di codifica BCH per il DVB-S2

1.3.1 Descrizione dell'algoritmo di codifica

La codifica dei messaggi può essere di due tipi: sistematica e non sistematica. Quando i dati vengono codificati in maniera sistematica i bit di messaggio vengono esattamente replicati all'interno della parola di codice. A questi vengono poi aggiunti alcuni bit di ridondanza, talvolta chiamati bit di parità. Nello standard DVB-S2 la ridondanza del codice occupa i bit meno significativi, mentre l'informazione quelli più significativi (si veda l'espressione (4.1)). Da ciò discende che i bit d'informazione occupano le posizioni più significative della parola di codice, cosa che tipicamente si verifica.

Il codice BCH, facendo parte della famiglia dei codici ciclici, condivide con essi l'algoritmo di codifica. Questo algoritmo consiste nella moltiplicazione del messaggio (espresso in notazione polinomiale) per il polinomio generatore del codice, se intendiamo avere una codifica non sistematica; se intendiamo invece ottenere una codifica sistematica, l'algoritmo consiste nella divisione del messaggio – riempito di un numero di zeri (*zero padding*) pari al numero di bit di ridondanza nelle sue posizioni meno significative – per il polinomio generatore, il cui resto rappresenta i bit di parità (o di ridondanza). La rappresentazione matematica del processo di codifica appena descritto è contenuta nella formula (4.5).

Le semplici architetture che realizzano l'operazione di codifica (tranne l'operazione di zero padding) sono chiamate *Linear Feedback Shift Register* (LFSR), poiché consistono in un registro a scorrimento con anello di retroazione (si veda la figura 4.1). Come è facile intuire, occorrerà disporre in fase di codifica di una logica di controllo, poiché questi semplici oggetti hanno la peculiarità di avere una risposta all'impulso infinita, ossia sono dei filtri *Infinite Impulse Response* (IIR).

Per quanto riguarda il loro funzionamento, i bit di messaggio entrano, dopo aver superato un nodo di somma (modulo 2), nella prima cella del registro a scorrimento. E' possibile provare che, dopo un certo numero di iterazioni, lo LFSR contiene all'interno del suo registro a scorrimento il resto della divisione del messaggio per il polinomio generatore del codice BCH (ciò è valido, più in generale, per un qualsiasi codice ciclico).

L'analisi del funzionamento di questo LFSR (figura 4.1) conduce alle seguenti osservazioni:

1. il dispositivo è in grado di calcolare il resto della divisione in un numero di iterazioni pari alla lunghezza della parola di codice.

2. il 'recupero' dei bit di ridondanza calcolati dall'LFSR, almeno seguendo la procedura finora descritta, può essere problematico e, comunque, deve essere effettuato in maniera non seriale, a meno di non spezzare l'anello di retroazione. Come già accennato, il problema è intrinsecamente legato all'architettura degli LFSR, che sono dei filtri IIR. Il problema potrebbe essere risolto 'congelando' lo stato del dispositivo per il tempo necessario all'estrazione dei bit di ridondanza dal registro a scorrimento.

Alcune considerazioni di natura matematica e architetturale fanno intuire la convenienza di fare entrare i bit di messaggio nel nodo di somma (modulo 2) insieme ai valori contenuti nell'ultima cella del registro a scorrimento: si noterà, in particolare, che questa modifica consentirà di risparmiare un numero di iterazioni pari al numero di bit di ridondanza, cioè pari al numero di celle del registro a scorrimento.

1.3.2 Architetture seriali

Cercando di riassumere, abbiamo quindi

- Una architettura seriale LFSR (figura 4.1) in grado di calcolare i bit di ridondanza (o di resto) del messaggio in un numero di iterazioni pari alla lunghezza della parola di codice. Il problema di questa architettura è che la difficoltà di estrazione dei bit di resto calcolati ne impedisce il funzionamento in modalità seriale e continua. Nonostante ciò, questa architettura sarà presa in considerazione nel successivo passo di parallelizzazione del sistema.

- Una architettura seriale LFSR (figura 4.2) in grado di calcolare i bit di resto del messaggio in un numero di iterazioni pari alla lunghezza del messaggio stesso, risparmiando in questa maniera un numero di colpi di clock pari al numero di celle del registro a scorrimento (uguali al numero di bit di ridondanza per ovvie ragioni). Si intuisce, inoltre, che questa architettura dovrebbe essere in grado di lavorare, almeno in via di principio, in modalità seriale e continua.

Aggiungendo all'architettura in figura 4.2 una logica di controllo che consenta, una volta calcolati i bit di resto (la cui rappresentazione polinomiale è data dalla espressione (4.3)), di spezzare l'anello di retroazione e di leggere serialmente i dati contenuti nelle celle del registro a scorrimento, si ottiene infine l'architettura di un codificatore seriale per un codice BCH sistematico.

Segue, ora, una breve descrizione del funzionamento di questo codificatore (figura 4.2) in grado di fornire la codifica sistematica dei messaggi in un numero di iterazioni pari alla lunghezza delle parole di codice e di operare in maniera seriale e continua. Dividiamo il processo di codifica in due parti

- **La trasmissione dei bit di messaggio e il calcolo diretto dei bit di resto.** Inizialmente e per un numero di colpi di clock pari al numero dei bit componenti il messaggio da codificare, i bit di messaggio vengono trasmessi, in accordo con le specifiche del DVB-S2, dal più significativo al meno significativo, alimentando al tempo stesso il registro a scorrimento (anello di retroazione chiuso). Questa modalità ad anello chiuso è necessaria per il calcolo dei bit di resto, raccolti nelle celle del registro a scorrimento.

- **Trasmissione dei bit di resto (o di ridondanza).** Successivamente, e per un numero di colpi di clock pari al numero dei bit di resto, l'anello di retroazione rimane aperto, dando modo così ai bit di ridondanza di poter uscire serialmente dalle celle del registro a scorrimento.

In conclusione, il tipo di architettura appena descritto realizza la divisione del messaggio per il polinomio generatore in un numero ridotto di colpi di clock, è in grado di operare in maniera realmente seriale e continua (nel senso che una volta codificato un messaggio è possibile modificarne immediatamente un altro), mantiene le caratteristiche di bassa complessità tipiche degli LFSR.

1.3.3 Algoritmi di codifica per architetture parallele

Dal punto di vista hardware, la lunghezza delle parole di codice (previste dal DVB-S2), i vincoli di frequenza imposti dell'analisi globale della sezione di trasmissione e la definizione dell'interfaccia con il codificatore LDPC che ha un ingresso parallelo, rendono necessario lo sviluppo di un modello di codificatore BCH parallelo, che sia

agevolmente integrabile con l'intera sezione di trasmissione del DVB-S2. In generale, le architetture parallele consentono di ottenere dei *throughput* più elevati e spesso vengono scelte per la realizzazione in hardware.

Dalla teoria del controllo, sappiamo che ogni sistema lineare può essere modellato attraverso un sistema di equazioni di stato (4.6). In questo tipo di rappresentazione matematica una equazione rappresenta l'evoluzione temporale dello stato del sistema; un'altra invece le uscite in funzione degli ingressi e dello stato per ogni istante di tempo. Il modello (del tutto generale) è stato particolarizzato per gli specifici sistemi (analizzati nella sezione precedente,) che realizzano la codifica BCH sistematica (si veda l'espressione (4.7)), concentrandoci solamente sull'equazione (4.7b) modellante l'evoluzione della traiettoria del vettore di stato, poiché esso rappresenta il resto dell'operazione di divisione del messaggio per il generatore polinomiale.

Riferendoci all'equazione (4.7b) che rappresenta l'evoluzione dello stato, ossia del contenuto delle celle del registro a scorrimento, definiamo la matrice di transizione di stato $\mathbf{A}$ e il vettore di trasferimento dell'ingresso sullo stato $\mathbf{b}$. Si osserva che, nel passaggio da una architettura seriale all'altra, le caratteristiche (si veda l'espressione (4.9)) della matrice di transizione di stato $\mathbf{A}$ rimangono, come era attendibile, invariate. Il vettore di trasferimento $\mathbf{b}$ invece subisce delle modifiche (si confrontino le espressioni (4.10) e (4.9)), poiché nell'architettura seriale più veloce c'è una più immediata incidenza degli ingressi sullo stato del sistema, poiché i bit in ingresso entrano, attraverso il nodo di somma, direttamente sul principale collegamento di retroazione.

Attraverso la definizione matematica delle due architetture seriali, mediante alcune semplici manipolazioni della equazione (4.7b), si perviene a un modello di sistema lineare a ingressi multipli (parallelo) e alla relativa equazione di stato. Il vettore di trasferimento si trasforma in una matrice di trasferimento $\mathbf{B}_p$, le cui caratteristiche dipendono sia dal livello di parallelismo prescelto sia dal tipo di architettura seriale che si è resa parallela o, più precisamente, dalla posizione dell'ingresso/i rispetto allo LFSR. La matrice di transizione di stato del sistema parallelo $\mathbf{A}$ non dipende dal tipo di architettura seriale da rendere parallela, ma solamente dal livello di parallelismo: infatti diventa $\mathbf{A}^p$, ossia la matrice del sistema seriale elevata a potenza.

1.4 Realizzazione hardware del codificatore BCH

Nella fase di realizzazione in hardware del dispositivo si è dovuto tenere conto delle seguenti condizioni relative all'architettura del codificatore LDPC e, più in generale, della intera sezione di trasmissione del DVB-S2 progettata e sviluppata da TAS-I.

- La memoria d'ingresso del codificatore LDPC ha un livello di parallelismo pari a 360 e quindi il livello di parallelismo del codificatore BHC dovrebbe essere un divisore di 360.

- Il livello di parallelismo del BCH dovrebbe, inoltre, poter essere un divisore delle lunghezze di blocco previste dal DVB-S2

- Il codificatore BCH dovrebbe avere un livello di parallelismo tale da evitare che costituisca un vero e proprio collo di bottiglia dell'intera catena di trasmissione del DVB-S2 sviluppata.

Per queste ragioni, nella realizzazione in hardware del codificatore BCH è stato scelto un livello di parallelismo pari a 8.

Inoltre, tra le due architetture parallele descritte nel capitolo 3 è stata scelta quella che impiega un numero inferiori di colpi di clock per la codifica dei messaggi (o blocchi di bit di informazione). Questo per avere, come verrà illustrato nella sezione dedicata al problema relativo al possibile cambio del rapporto di codifica tra una trama e l'altra, una architettura in grado di avere delle caratteristiche dinamiche e una versatilità maggiori (rispetto, ovviamente, all'architettura parallela corrispondente a quella in figura 4.1).

1.4.1 Descrizione del codificatore

L'architettura hardware (illustrata in figura 5.4) del codificatore BCH comprende un registro a 192 celle (*Flip Flop*), 192 reti combinatorie d'ingresso e 192 reti combinatorie d'uscita, pilotate opportunamente da coefficienti (precalcolati) memorizzati nelle LUT, e alcuni sommatori, che, per le note proprietà dell'algebra binaria, sono delle porte XOR a due ingressi. Il codificatore BCH compie per ogni processo di codifica le seguenti operazioni:

- I bit informativi, che arrivano in parallelo dal bit meno significativo al più significativo, vengono processati dalle 192 logiche combinatorie di ingresso, che hanno il compito di realizzare il prodotto (riga-colonna) tra la matrice $\mathbf{B}_p$ e gli ingressi, come illustrato in figura 5.3.

- Le reti combinatorie più vicine all'interfaccia con il codificatore LDPC (figura 5.2) processano invece gli ultimi otto (o, più in generale, p) bit, realizzando in gran parte il processo di retroazione. Esse, come per le reti combinatorie agenti sugli ingressi, realizzano gran parte del prodotto (si veda l'equazione (5.1) e si osservi la struttura della matrice $\mathbf{A}^p$ in (4.15)) tra la matrice $\mathbf{A}^p$ e il vettore di stato.

- Le funzioni logiche relative alla sottomatrice diagonale $\mathbf{I}$ contenuta in $\mathbf{A}^p$ sono realizzate dalle porte XOR a due ingressi (nodi di somma su GF(2)) connesse all'uscita (come illustrato nella figura 5.2) di alcune delle reti combinatorie agenti sugli ultimi 8 bit di stato.

- Al termine del numero di clock richiesti (in merito si consulti la tabella 5.1) per il calcolo dei bit di resto, il contenuto del registro deve essere azzerato prima di poter procedere con la codifica del messaggio successivo.

Come appena mostrato, le due reti combinatorie sono in realtà delle porte XOR a otto ingressi, le cui abilitazioni sono programmate dai coefficienti delle matrici facilmente memorizzabili in delle Look Up Table (LUT).

L'architettura finora mostrata realizza in realtà la codifica BCH solamente per il livello di protezione più elevato (cioè $t = 12$) previsto dallo standard. Come accennato nell'introduzione, il DVB-S2 prevede che il codificatore possa variare il suo livello di codifica trama per trama. Perciò occorre avere una architettura hardware sufficientemente flessibile, in grado di poter assicurare la realizzazione di questa caratteristica. Sebbene il generatore polinomiale del codice vari a seconda della sua capacità correttiva, la struttura della matrice $\mathbf{A}^8$, legata specialmente all'architettura dello LFSR, rimane invariata. Tuttavia le dimensioni di $\mathbf{A}^8$ diminuiscono e, contestualmente, variano i suoi coefficienti poiché, come appena detto, varia il polinomio generatore del codice. Il problema dei coefficienti viene facilmente superato, memorizzando gli stessi in delle LUT (*Look Up Table* o tabelle di riferimento); il problema relativo al calo delle dimensioni delle matrici $\mathbf{A}^8$ e $\mathbf{B}_8$, invece, può essere risolto, inserendo queste matrici all'interno di matrici sovradimensionate (cioè di dimensioni pari a quelle corrispondenti alla capacità correttiva massima) come mostrato in (5.3). Questa operazione descritta in termini matematico matriciali corrisponde all'inibizione delle prime 32 ($t = 10$) o 64 ($t = 8$) coppie di reti combinatorie per mezzo dei coefficienti nulli memorizzati nelle LUT.

Alla luce di queste nuove considerazioni, circa le proprietà dinamiche di cui dovrebbe godere il codificatore BCH per DVB-S2, vogliamo ora giustificare la scelta di questa architettura hardware piuttosto che quella più lenta (almeno a livello teorico), la cui realizzazione avrebbe portato a risparmiare le 192 logiche combinatorie d'ingresso utilizzate dall'architettura da noi scelta. Senza entrare in maggiori dettagli, il motivo principale risiede proprio nella migliore facilità di realizzare un codificatore in grado di rispondere all'esigenza di poter cambiare la capacità correttiva del codice al termine di ogni processo di codifica.

1.4.2 Descrizione dell'interfaccia BCH-LDPC

Passiamo ora a descrivere il funzionamento dell'interfaccia BCH-LDPC, che ha il compito di passare al codificatore a valle la parte sistematica del codice BCH più i bit di ridondanza nell'ordine e formato compatibile con il DVB-S2. La figura 5.5 mostra l'architettura dell'interfaccia composta da:

- un registro a scorrimento (figura 5.6) su cui vengono salvati i 192 elementi (bit

di ridondanza) del registro del codificatore in un solo colpo di clock (caricamento parallelo), che successivamente vengono trasmessi su un bus a 8 bit. Il numero di colpi di clock richiesti (i valori sono riportati in tabella 5.1) per la completa lettura dei bit di ridondanza contenuti nel registro a scorrimento dipende dalla modalità operativa (per quanto riguarda la FEC) del DVB-S2;

- un multiplexer a cui è connesso sia il bus in uscita del suddetto registro a scorrimento sia il bus che trasporta i bit sistematici, che alimentano il codificatore BCH.

Segue l'ordine temporale delle operazioni effettuate da questa interfaccia.

1. Una volta calcolati, i bit di ridondanza vengono salvati in un registro a scorrimento secondo l'associazione mostrata in figura 5.6. Per il caricamento parallelo del registro a scorrimento ci si serve di alcuni multiplexer, inserendoli tra le sue celle.

2. Ora i bit appena salvati, durante il loro scorrimento attraverso le celle del registro, vengono 'catturati' da alcune linee che li convogliano sul bus di uscita a 8 bit.

3. Mediante una opportuna logica di controllo e il multiplexer che precede l'LDPC, dapprima i bit sistematici e, successivamente, i bit di ridondanza vengono scritti nella memoria del codificatore LDPC.

1.5 Pacchetto software per la validazione del modello VHDL

1.5.1 Implementazione software del codificatore seriale

La funzione C sviluppata emula il comportamento del codificatore BCH seriale mostrato in figura 4.2. A tal fine sono state utilizzate le seguenti variabili e parametri:

- il parametro `ticks`, che si riferisce al numero di iterazioni o colpi di clock che si desidera simulare;

- il vettore `m[]` da cui vengono letti i bit di messaggio e il vettore `out[]` su cui vengono scritti i bit in uscita del codificatore;

- il vettore `state[]`, che contiene i bit del registro a scorrimento;

- il contatore `encStep`, che indica il numero complessivo di iterazioni compiute dal codificatore. Quando questo contatore supera il numero di bit informativi, i bit di ridondanza calcolati dal codificatore vengono scritti nel vettore di uscita `out[]`;
- il vettore `g[]`, che contiene i coefficienti del polinomio generatore del codice BCH;
- la variabile `r`, che indica la lunghezza del registro a scorrimento o, in altre parole, il numero di bit di ridondanza.

Vale la pena di sottolineare che nulla impedisce di adoperare la funzione `Run()` della classe `BCHenc` per simulare il funzionamento del codificatore parallelo dato che il parametro `ticks` è completamente definibile dall'utente.

1.5.2 Implementazione software del codificatore parallelo

I coefficienti delle matrici $\mathbf{A}^8$ e $\mathbf{B}_8$, calcolati per mezzo di apposite routine Matlab, sono stati salvati in alcune variabili locali. Più precisamente, le variabili locali contengono le sole sottomatrici $\mathbf{C}_1$ e $\mathbf{C}_2$ riferite alle reti combinatorie di uscita e le matici $\mathbf{B}_8$, una per ogni polinomio generatore previsto dal DVB-S2.

Per quanto riguarda le reti combinatorie, sono state utilizzate delle funzioni ausiliarie che ne imitano il comportamento in hardware

- la funzione `combc(index, input[])`, che realizza per ogni rete combinatoria (rappresentata dal parametro `index`) il prodotto riga per colonna tra la matrice $\mathbf{B}_8$ e il vettore di ingresso;
- la funzione `combk(index,regold)`, che realizza il prodotto riga per colonna tra le sottomatrici $\mathbf{C}_1$ e $\mathbf{C}_2$ e $x_{184}, \ldots, x_{191}$, gli ultimi otto bit del vettore di stato.

La parte iniziale della funzione `BCHkclkpar(n,k)` sviluppata carica dalla memoria le matrici corrispondenti alla coppia (n, k) associata alle modalità di codifica del DVB-S2. Il ciclo `for` nella parte centrale dell'algoritmo aggiorna ciclicamente il registro di stato del codificatore BCH. Infine, i bit calcolati vengono disposti in maniera conforme allo standard DVB-S2 e al formato di codifica sistematica.

1.5.3 Tabelle dei campi di Galois

Come accennato nel capitolo 3, la conoscenza del campo di Galois GF(2^{16}) in cui giacciono le radici dei polinomi generatori dei codici BCH è necessaria per la decodifica e/o la rilevazione degli errori occorsi in trasmissione. Poiché il BCH del DVB-S2

è primitivo, il campo di Galois ad esso associato può essere costruito utilizzando il polinomio $g_1(x)$ nella tabella 3.4. Tale polinomio è infatti il polinomio primitivo di GF(2^{16}) per le ragioni indicate nel capitolo 3.

I $2^{16} - 1$ elementi di GF(2^{16}) possono essere ottenuti attraverso una struttura a LFSR le cui interconnesioni sono stabilite dai coefficienti del polinomio primitivo del campo. Se questo LFSR viene inizializzato con $(0, \ldots, 0, 1)$ l'evoluzione del suo stato, in assenza di stimoli esterni, si ripete ogni $2^{16} - 1$ iterazioni: ciò significa che il valore con cui abbiamo inizializzato le celle del registro a scorrimento è un elemento primitivo di GF(2^{16}), che, come tale, può essere utilizzato nella costruzione di utili tabelle grazie alle quali è possibile eseguire facilmente le operazioni di prodotto tra gli elementi di GF(2^{16}). Nelle nostre simulazioni è stato quindi utilizzato $\alpha = 1$ (in notazione decimale) come elemento primitivo.

Ad ogni iterazione dell'algoritmo, che simula il funzionamento dell'LFSR in assenza di stimoli esterni, vengono costruite e salvate le seguenti due tabelle:

- `powOfAlpha[i]`, che rappresenta α^i per $i = 0, \ldots, 2^{16} - 2$
- `indexAlpha[i]`, che rappresenta il logaritmo in base α di un generico elemento del campo GF(2^{16})

1.5.4 Decodifica BCH

La decodifica algebrica di un codice BCH ad alfabeto binario consta dei seguenti passi:

- **Calcolo della sindrome.** L'operazione consiste nella valutazione delle parole di codice nelle $2t$ radici del polinomio generatore come indicato nell'equazione (6.1). Questo calcolo è realizzato dalla funzione `errordetection()`, che utilizza le tabelle dei campi di Galois e gli operatori bit a bit del C. La funzione, inoltre, ritorna il valore logico vero (`true`) se la sindrome non è nulla, altrimenti ritorna un valore logico falso (`false`). In caso di rilevazione di errori si ricorre all'algoritmo di Berlekamp-Massey.
- **La determinazione del polinomio locatore degli errori**, le cui radici forniscono un'indicazione sulla posizione degli errori nella parola di codice. L'algoritmo di Peterson e quello di Berlekamp-Massey consentono di trovare il polinomio (6.4) locatore degli errori. Nel pacchetto software sviluppato è stato utilizzato l'algoritmo di Berlekamp-Massey per la determinazione dei coefficienti di (6.4). Questo algoritmo, sfruttando la relazione lineare (LFRS) (6.5) tra la sindrome e i coefficienti del polinomio locatore degli errori, costruisce iterativamente un LFSR in grado di riprodurre la sequenza $\{S_1, S_2, \ldots, S_{2t}\}$. I coefficienti di questo LFSR rappresentano i coefficienti del polinomio locatore degli errori.

- **La ricerca delle radici del polinomio locatore degli errori.** Generalmente questa operazione viene fatta usando la ricerca di Chien (*Chien search*). Essa consiste nella valutazione della (6.4) in tutti gli elementi del campo, in successione, diversi da zero. Tale operazione sia in hardware che in software non è molto costosa. Se le radici, trovate tramite la ricerca di Chien, sono distinte e giacciono nel campo di riferimento, allora possono essere usate per determinare la posizione degli errori. Viceversa, se le radici sono coincidenti e non giacciono nel campo di riferimento, allora la decodifica fallisce.

La funzione `BerlMass()` che realizza l'algoritmo di Berlekamp-Massey utilizza le seguenti variabili:

- i vettori/tabelle `pow` e `index`, che sono utili nelle operazioni di moltiplicazione e conversione in notazione esponenziale degli elementi del campo di Galois;
- il vettore `c[]`, che indica le interconnessioni dell'LFSR attuale;
- il vettore `p[]`, che indica le interconnesioni di un LFSR usato in precedenza;
- la variabile `d`, che contiene la discrepanza (cioè la differenza tra il valore desiderato e quello prodotto dall'LFSR corrente) calcolata all'istante corrente;
- la variabile `dm`, che indica la discrepanza calcolata quando l'LFSR prescelto aveva le interconnesioni salvate nel vettore `p[]`;
- un vettore ausiliario `T[]`, usato durante la routine d'aggiornamento del vettore `c[]`;
- la variabile `l`, che rappresenta la quantità di scalamento da applicare durante l'aggiornamento;
- la variabile `k`, che conteggia le iterazioni dell'algoritmo;
- la variabile `L`, che contiene la lunghezza attuale dell'LFSR.

Per concludere, seguono i passi principali dell'algoritmo, che dopo $2t$ iterazioni trova i coefficienti del polinomio locatore degli errori.

- Calcolo della discrepanza attraverso la formula $S_k + \sum_{i=i}^{L} c_i S_{k-i}$. Questo calcolo è facilmente realizzato utilizzando le tabelle dei campi e gli operatori bit a bit.
- Se la discrepanza è nulla, allora non occorre effettuare modifiche allo LFSR corrente (`l` viene incrementato di una unità). Se invece la discrepanza non è nulla ci sono due alternative: se `2*L=>k`, `c[]` viene aggiornato, ma mantiene la sua lunghezza; se invece `2*L<k` allora variano sia il valore di `c[]` sia la sua lunghezza. Il vecchio valore di `c[]` viene salvato in `p[]` e la discrepanza associata in `dm`; la variabile `l` viene riportata a 1.

1.6 Prove di laboratorio preliminari della sezione TX del DVB-S2

1.6.1 Introduzione

In questa sezione vengono presentati i risultati ottenuti dalla preliminare campagna di test effettuata sulla sezione di trasmissione del DVB-S2 sviluppata da TAS-I. Il codice VHDL (*Very high speed integrated circuits Hardware Description Language*) è stato verificato in due passi: una prima verifica è stata fatta confrontando i risultati della simulazione RTL (*Register Transfer Logic*) con quelli ottenuti con il simulatore software sviluppato; una seconda verifica è stata poi effettuata analizzando i risultati ottenuti sintetizzando su una Stratix II il progetto e analizzando qualitativamente le costellazioni ottenute (figure 7.5, 7.7, 7.9). Di seguito vengono elencati e brevemente descritti i blocchi funzionali che costituiscono la sezione di trasmissione che è stata collaudata:

Interfaccia di ingresso È il blocco di interfaccia con il 'mondo esterno' e ha il compito di acquisire un numero adeguato di pacchetti MPEG che dipende dalla velocità di simbolo R_s, dalla modulazione e dal tasso di codifica del PLFRAME che deve essere generato. Il blocco svolge funzioni legate all'applicazione, fornisce l'interfaccia per il flusso di ingresso e gli strumenti opzionali richiesti per l'ACM ; inserisce la codifica CRC (*Cyclic Redundancy Check*) per permettere al ricevitore di rivelare la presenza di errori nel flusso ricevuto.

Formattatore DVB-S2 Questo blocco compone la trama per il formato FECFRAME normale.

Inserimento della intestazione Questo blocco inserisce una intestazione di lunghezza fissa (BBHEADER) di 10 Byte (80 bit) davanti al campo dati per informare il ricevitore del formato del flusso di ingresso e del tipo di "adattamento" utilizzato. Nel caso i dati utente, disponibili per la trasmissione, non siano sufficienti a riempire completamente il BBFRAME, si provvede a completarlo con zeri di riempimento (*zero padding*).

BB Scrambling Per evitare la presenza di sequenze critiche per il codice FEC, il BBFRAME viene moltiplicato per una sequenza pseudocasuale generata da un LFSR e inizializzata all'inizio di ogni trama. La sequenza di *scrambling* è memorizzata in una ROM e i bit informativi vengono processati in parallelo.

Physical Layer Scrambling Block Blocco simile al precedente, ma che agisce a livello fisico, che rende casuale la sequenza (complessa) prima che entri nel modulatore.

Blocco di codifica Questo blocco effettua la codifica concatenata del codice esterno (BCH) e interno (LDPC).

Blocco di interlacciamento dei bit La diretta implementazione del metodo di interlacciamento dei bit suggerito da Hughes Network Systems (HNS) riduce la velocità di simbolo massima raggiungibile, che è pari a f_{ck}/η. Attraverso l'uso, invece, di una architettura parallela, capace di leggere η bit consecutivi, appartenenti a simboli differenti, per ogni colpo di clock, la velocità di simbolo è diventata pari a f_{ck}.

Blocco di mapping Associa i bit alla costellazione QPSK, 8-PSK, 16-ASK o 32-APSK a seconda delle applicazioni. Il *mapping* delle costellazioni è memorizzato in una ROM. Le componenti I e Q sono entrambe rappresentate su 7 bit.

Blocco di generazione della trama PL Gestisce l'inserzione dell'intestazione di livello fisico e dei simboli pilota opzionali, di PLFRAME fittizi (Dummy Frame) in assenza di dati utili pronti per la trasmissione, e la moltiplicazione per una sequenza pseudocasuale (Scrambler) per la dispersione dell'energia.

Sezione di modulazione In questa sezione si applicano il filtraggio in banda base e la modulazione in quadratura per modellare lo spettro del segnale e per generare il segnale a radiofrequenza. Il filtro usato in trasmissione è il tipico filtro a radice di coseno rialzato (SRRC, *Squared Root Raised Cosine*) con tre possibili coefficienti di *roll-off* (0,35, 0,25 e 0,20, per soddisfare i diversi requisiti di occupazione di banda).

Sezione di interpolazione Interpolatore di Farrow del terzo ordine utilizzato per modificare il rate in uscita del filtro polifase.

Sezione di conversione a frequenza intermedia Questo blocco effettua una conversione in frequenza: porta il segnale in banda base a frequenza intermedia.

Filtro di precompensazione del DAC È un filtro FIR che compensa la distorsione (circa 4 dB dalla continua alla metà della frequenza di campionamento) dovuta alla conversione digitale analogica del segnale.

1.6.2 Setup di misura

Il banco di misura allestito per l'esecuzione dei test è dato in figura 7.2 e comprende:

1. Scheda di sviluppo DSP Stratix II

2. Oscilloscopio Agilent
3. SW VSA (Vector Signal Analyzer) istallato su PC portatile connesso all'oscilloscopio per l'acquisizione dei dati tramite porta ETHERNET.

La scheda di sviluppo include una Stratix II EP2S180, utilizzata per la sintesi del progetto VHDL e due convertitori DAC a 14 bit (165 Msample/s). Solo uno dei due convertitori è stato utilizzato per la conversione del segnale e l'uscita del DAC collegata all'oscilloscopio. I dati acquisiti da quest'ultimo sono stati inviati tramite porta ETHERNET al PC e acquisiti dallo stesso per essere processati dal VSA.

L'analisi statistica fornita dal software ha rappresentato un utile strumento di valutazione qualitativa delle prestazioni del trasmettitore in termini non solo di qualità delle costellazioni (per semplice ispezione visiva), ma anche in termini di valore medio degli errori di ampiezza e fase della costellazioni. In particolare, oltre all'errore sull'ampiezza e sulla fase, viene calcolata l'ampiezza del vettore errore (EVM, *Error Vector Magnitude*) definito come il vettore congiungente il punto della costellazione atteso e quello ricevuto (si osservino le figure 7.3 e 7.4).

1.6.3 Risultato del test

Considerando le risorse HW a disposizione in termini di massima velocità di campionamento del DAC (165 Msample/s) e caratteristiche del filtro anti-aliasing all'uscita del DAC, i test sono stati eseguiti sui seguenti segnali: 2Mbaud 16 APSK e 30 MBaud 8 PSK e 16 APSK. Le costellazioni risultano essere 'compatte' e gli errori in ricezione presentano un valor medio piuttosto ridotto; inoltre è evidente l'effetto di distorsione introdotto dal DAC in figura 7.7b (si osserva una costellazione piuttosto *scattered*).

1.7 Conclusioni

Nel documento di tesi si discute il progetto di una sezione di trasmissione numerica che assicura una trasmissione affidabile delle informazioni via satellite, rispettando i requisiti sfidanti sulle prestazioni imposti dallo standard DVB-S2.

Il mio impegno e partecipazione al progetto e allo sviluppo di questa sezione di trasmissione si è specialmente concentrato su:

- lo studio delle proprietà generali dei codici BCH, utili nell'analisi della struttura del codice BCH adoperato dallo standard DVB-S2 e nella valutazione degli algoritmi di codifica e decodifica;
- l'analisi teorica del codice BCH del DVB-S2 per mezzo della quale si è giunti alla conclusione che esso sia primitivo e *shortened*;

- la modellazione e il progetto per un codificatore BCH ad alto livello di parallelismo in modo da rispettare le specifiche progettuali imposte dalla sezione di trasmissione;
- lo sviluppo in linguaggio C dell'algoritmo di decodifica algebrica di Berlekamp-Massey in modo da poter verificare le capacità di correzione degli errori fornite dalla codifica BCH; a questo scopo sono state inoltre sviluppate delle funzioni in grado di calcolare e memorizzare le tabelle dei campi di Galois;
- lo sviluppo di un pacchetto software per verificare il corretto funzionamento degli algoritmi e architetture di codifica BCH scelti e dare supporto ai progettisti VLSI (*Very Large Scale Integration*) durante lo sviluppo e la sintesi del modello VHDL (*Very high speed integrated circuits Hardware Description Language*).

Chapter 2

Applications and Features of Satellite Communications

2.1 Introduction

Satellite communications provide unique solutions for a number of communications and strategic applications that are possible only using the satellite environment. These applications are typically 'nice' applications, such as:

- Multimedia communications provided to wide geographical areas with low population density; in this case the deployment of the satellite is less expensive than the corresponding terrestrial network to provide the same service;
- Maritime communications (e.g. Inmarsat) also associated to radio-navigation systems;
- Television broadcasting: in this case, a single signal is broadcasted for a wide community of users, making more economically efficient the use of the satellite when compared to a wide network of local broadcasting stations to cover the geographical area;
- Earth observation and monitoring systems offering a distinctive potential for strategic (military and commercial) information gathering.

All of the above applications require from medium to high data rate communication links, which need to be implemented with the resources typical of a spacecraft. Clearly the onboard available power, weight and antenna size are limited: this leads to the requirement of promoting state-of-the art research in the field of modulation and coding, as well as synchronization techniques able to operate al close as possible to the ultimate limits of performances, at very low signal to noise ratios and saving

as much as possible the used bandwidth, which, besides, has to be shared among various coexisting satellites as well as terrestrial applications.

Hence state-of-the-art satellite communications for Earth Observation and Telecommunications (which are among the most appealing applications requiring high data rate modems) are characterized by the need for both maximum information throughput and minimum power and bandwidth consumption, requirements evidently contradictory with an inter-relation theoretically stated by the Shannon Bound. Additionally the available bandwidth, data rate and signal to noise ratio specifications vary according to each specific mission/from one mission to another. An ideal all-purpose MODEM unit, flexible for any mission application and hence commercially appealing, should have the following features:

- Flexible configurability of data rate and bandwidth allocated;
- High variability of signal to noise ratio of operation;
- Performance always at 1 dB maximum from the Shannon Capacity Bound (fixed modulation and coding quality);
- Efficient usage of power and hardware resources for processing.

Such a modem would allow having unique off-the-shelf solution, matching the needs of almost all the missions for Earth Observation and extra-high-speed telecommunications.

The novel DVB-S2 modulation and coding scheme as well as the Modem for Higher Order Modulation Schemes (MHOMS) performance are so close to the Shannon Bound that they are expected to set the modulation and coding baseline for many years to come. The MHOMS program was financed by European Space Agency (ESA). The envisaged MHOMS modem application scenarios will encompass as a minimum:

- High-speed distributed Internet access;
- Trunk connectivity (backbone/backhaul);
- Earth observation high-speed downlink;
- Point-to-multipoint applications (e.g., high-speed multicasting/broadcasting).

Further details on MHOMS are available in [19, 4].

2.2 Remote Sensing and Earth Observation

Earth observation satellites have a key role to play in both the civilian and military sectors, monitoring nearly all aspects of our planet. This is the only technology capable of providing truly global coverage, particularly over the vast expanses of the oceans and sparsely populated land areas (e.g. deserts, mountains, forests, and polar regions). Earth observation data is being used by more than 300 research teams. Small high-tech firms, large corporations and public agencies (meteorological offices, etc.) use Earth-observation data for both operational and commercial purposes.

To provide professional services such as oceanography, meteorology, climatology, etc. to the end user, Earth observation satellites requires an high capacity of acquiring and processing a large volume of images on a daily basis. For example, COSMO-Sky-Med, in full constellation configuration, can acquire up to 560 GB per day, roughly corresponding to 1800 standard images.

COSMO-SkyMed interoperable system [20] is a remarkable example of a telecommunications integrated system capable of supporting the ability to exchange data with other external observation systems according to agreed modalities and standard protocols. The COSMO-SkyMed (Constellation of four Small Satellites for Mediterranean basin Observation) is a low-orbit, dual-use Earth observation satellite system operating in the X-band. It is funded by the Italian Ministry of Research (MIUR) and Ministry of Defense and managed by the Italian space agency (ASI). COSMO-SkyMed 1 has been successfully launched on June 7th, 2007 and COSMO-SkyMed 2 on December 10, 2007.

Thales Alenia Space is the program prime contractor, responsible for the development and construction of the entire COSMO-SkyMed system, including the space segment, with four satellites equipped with high resolution X-band synthetic aperture radars (SAR) and the ground segment, a sophisticated turnkey geographically distributed infrastructure. The ground segment is realized by Telespazio and includes the satellite control segment, mission planning segment, civilian user segment, defense user segment, mobile processing stations and programming cells. Thales Alenia Space is also responsible for the mission simulator, the integrated logistics and all operations up to delivery of the constellation.

Three particular aspects characterize COSMO-SkyMed's system performance: the space constellation's short revisit time for global Earth coverage; the short system response time; and the multi-mode SAR instruments imaging performance. COSMO-SkyMed features satellites using the most advanced remote sensing technology, with resolution of less than one meter, plus a complex and geographically distributed ground segment produced by Italian industry. COSMO-SkyMed's high image performance relies on the versatile SAR instrument, which can acquire images in different operative modes, and to generate image data with different (scene) size and resolution (from medium to very high) so as to cover a large span of applicative

needs.

Synthetic Aperture Radars (SARs) are most employed to gather a large amount of high resolution images. In this field, Thales Alenia Space played an important role in the development of the X-SAR (SAR operating in X band) space and ground segments. X-SAR is a joint project between NASA, the German space agency (DARA), and the Italian space agency (ASI) as part of NASA's "Mission to Planet Earth". X-SAR can measure virtually any region of the Earth in all weather and sunlight conditions, as well as penetrate vegetation, ice and dry sand, to map the surface and provide scientists with detailed information on climatic and geological processes, hydrological cycles and ocean circulation. X-SAR has flown on the U.S. Space Shuttle twice, in April and September 1994, both times on Endeavour. The two consecutive flights observed the same sites on Earth during two different seasons in order to detect any seasonal changes, thus contributing to more precise and reliable monitoring of renewable and nonrenewable resources. The SIR-C/X-SAR missions observed more than 400 sites. Based on the knowledge acquired during these missions, Thales Alenia Space participated in the Shuttle Radar Topography Mission (January 2000) which re-used the X-SAR radars to produce the first three-dimensional map of the Earth's surface.

Concerning the applications, the space constellations and versatility of the multimode SAR instrument enable a wide variety of image data products with different sizes, resolutions, elaboration levels, and accuracies (e.g. geo-localization), suitable for different applications needs, and allowing large environmental monitoring, territory surveillance, intelligence applications, and critical phenomena detection. By interferometric images, three-dimensional Digital Elevation Model products can be obtained and used to render a city with buildings, streets, or high-resolution topography.

COSMO-SkyMed mission specifications are compatible with numerous applications relating to civilian and military/security domains with 2D or 3D data. 2D applications are mainly related to cartographic purpose. Other applications are aimed at:

- Agricolture, for crop mapping and management;
- Landscape classification, for basic land use identification;
- Geology mapping for mineral, oil, and gas resources detection;
- Accurate urban area mapping for urban evolution surveys and management (identification of high-density and low-density urban zones), as well as industrial zone mapping with identification of buildings of interest, port implantations, storage zones, airports, etc.;

- Road and network (electricity, gas, telecom) identification and mapping, for compiling the Geographic Information System (GIS) data base (note, e.g., its relevance for applications such as radio navigation and safety of life services);

- Survey mapping.

Other applications fields concern hydrology, water resource inventory and detection, coastal zones for erosion and coastal line determination. 3D possible applications are various:

- *Decision help*: based on 3D models as powerful tools. They allow citizens, city authorities and managements services to understand and explore a territory, and to build up a database for realistic simulations. For example, architecture, urbanization and new transport infrastructures can be simulated via a 3D database and presented to the different city players;

- *Natural and industrial risk management*: simulation, prevention and post-crisis. Using a Digital Elevation Model combined with a high-resolution image allows phenomena simulation such as flood, fire, earthquake, air pollution, etc. It can be used during different risk management phases, such as:

  - Prevention: preparation of intervention missions, reports of risky areas to survey, simulation of phenomena and related interventions;
  - Crisis anticipation: types of risky building, communication network failure, secondary itineraries to use, 3D localization of interesting positions (e.g. helicopter landing zones);
  - Post-crisis: damage evaluation, reconstruction planning.

In mid-2007 Italy and France have been activated a bilateral image-trading protocol providing for reciprocal access to Helios 2 and COSMO-SkyMed data. This agreement enabled the French and Italian armed forces to benefit from the complementary optical and radar image capabilities of these systems.

This complex, challenging system delivers outstanding security, international cooperation (for civilians and, within NATO countries, for defense), scalability and interoperability. In other words, COSMO-SkyMed provides Italy and partner countries with one of the worldŠs most technologically advanced observation systems to guarantee greater security and an improved standard of living.

2.3 The Second Generation of Digital Video Broadcasting System

DVB-S2 is the second-generation DVB specification for broadband satellite applications, developed on the success of the first generation specifications (i.e. DVB-S for broadcasting and DVB-DSNG for satellite news gathering and contribution services), benefiting from the technological achievements of the last decade.

The DVB-S2 system has been designed for various broadband satellite applications

- broadcasting services of Standard Definition TeleVision (SDTV) and High Definition TeleVision (HDTV);
- interactive applications oriented to professional or domestic services (i.e. Interactive Services including Internet Access for consumer applications);
- professional service of Satellite News Gathering (SNG);
- distribution of TV signals (VHF/UHF) to Earth digital transmitters;
- data distribution and Internet Trunking.

The DVB-S2 standard achieves an higher capacity of data transmission than the first generation systems, an enhanced flexibility, a reasonable complexity [1] of the receiver. In fact, it has been specified around three key concepts: best transmission performance, total flexibility and reasonable receiver complexity.

To obtain a well balanced ratio between complexity and performance, DVB-S2 benefits from more recent developments in channel coding and modulation. Employment of these novel techniques translates in a 30 percent of capacity increase over DVB-S under the same transmission conditions.

To achieve the top quality performance, DVB-S2 is based on LDPC (Low Density Parity Check) codes, simple block codes with very limited algebraic structure, discovered by R. Gallager in 1962. LDPC codes have an easily parallelizable decoding algorithm which consists of simple operations such as addition, comparison and table look-up; moreover the degree of parallelism is adjustable which makes it easy to trade-off throughput and complexity[2]. Their key characteristics, allowing quasi-error free operation at only 0,6 to 1,2 dB [6] from the Shannon limit, are:

[1] More precisely, the complexity is referred to amount of total operations to perform, but it is a quantity difficult to define, since it is strongly dependent on the kind of implementation (analogic, digital) either implemented in hardware or software and on the selected design options.

[2] For the sake of truth it has to be highlighted that, after the DVB-S2 contest for modulation and coding was closed, other coding schemes have been reconsidered by TAS-I for extra high speed operations, with less decoding complexity and almost equal performance. This has been made possible emulating the LDPC parallelism via ad-hoc designed parallel interleavers.

- the very large LDPC code block length, 64 800 bits for the normal frame, and 16 200 bits for the short frame (remind that block size is in general related to performance);

- the large number of decoding iterations (around 50 Soft-Input-Soft-Output (SISO) iterations);

- the presence of a concatenated BCH outer code (without any interleaver), defined by the designers as a "cheap insurance against unwanted error floors at high C/N ratios".

Variable Coding and Modulation (VCM) functionalities allows different modulation and error protection levels which can be switched frame by frame. The adoption of a return channel conveying the link conditions allows to achieve closed-loop Adaptive Coding and Modulation (ACM).

High DVB-S2 flexibility allows to deal with any existing satellite transponder characteristics, with a large variety of spectrum efficiencies and associated C/N requirements. Furthermore, it is not limited to MPEG-2 video and audio source coding, but it is designed to handle a variety of audio-video and data formats including formats which the DVB Project is currently defining for future applications. Since video streaming applications are subjected to very strict time-constraints (around 100 ms for real-time applications), the major flexibility of present standard can be effectively exploited so as to employ *unequal error protection* [7, 14, 13] techniques and differentiated service level (i.e. priority in the delivery queues). This kind of techniques are devoted to obtain, even in the worst transmission conditions, a graceful degradation of received video quality. More details about these interesting unicast IP services are in [7].

Addressing into more details the possible applications and the relevant definitions, we observe that the DVB-S2 system has been optimized for the following broadband satellite application scenarios.

Broadcast Services (BS): digital multi-programme Television (TV)/High Definition Television (HDTV) broadcasting services. DVB-S2 is intended to provide Direct-To-Home (DTH) services for consumer Integrated Receiver Decoder (IRD), as well as collective antenna systems (Satellite Master Antenna Television - SMATV) and cable television head-end stations. DVB-S2 may be considered a successor to the current DVB-S standard, and may be introduced for new services and allow for a long-term migration. BS services are transported in MPEG Transport Stream format. VCM may be applied on multiple transport stream to achieve a differentiated error protection for different services (TV, HDTV, audio, multimedia).

Interactive Services (IS): interactive data services including internet access.

DVB-S2 is intended to provide interactive services to consumer IRDs and to personal computers. No recommendation is included in the DVB-S and DVB-S2 standards as far as the return path is concerned. Therefore, interactivity can be established either via terrestrial connection through telephone lines, or via satellite. Data services are transported in (single or multiple) Transport Stream format or in (single or multiple) generic stream format. DVB-S2 can provide Constant Coding and Modulation (CCM), or ACM, where each individual satellite receiving station controls the protection mode of the traffic addressed to it.

DTVC and DSNG: Digital TV Contribution and Satellite News Gathering. Digital television contribution applications by satellite consist of point-to-point or point-to-multipoint transmissions, connecting fixed or transportable uplink and receiving stations. Services are transported in single (or multiple) MPEG Transport Stream format. DVB-S2 can provide Constant Coding and Modulation (CCM), or Adaptive Coding and Modulation (ACM). In this latter case, a single satellite receiving station typically controls the protection mode of the full multiplex.

Chapter 3

DVB-S2 MODEM Concepts and Architecture

3.1 Modulation and Coding Schemes

The coding/modulation (C/M) problem consists of finding practical ways of communicating discrete messages reliably on a realistic channel: this may involve space and satellite communications, data transmission over twisted-pair telephone wires, shielded cable-TV wire, data storage, digital audio/video transmission, mobile communication, terrestrial radio, deep-space radio, indoor radio, or file transfer. The channel causes transmission quality degradation, as it includes attenuation, thermal noise, intersymbol interference, multiple-user interference, multipath propagation, and power limitations. The most general statement about the selection of a C/M scheme is that it should make the best possible use of the resources available for transmission, bandwidth, power, and complexity, in order to achieve a required Quality of Service (QoS), i.e, the specified minimum performance requirement imposed on the service. Typically, the latter is expressed in terms of error probability, which is a decreasing function of the signal-to-noise ratio (SNR). Sensible strategies for C/M design must define a trade-off among four factors:

- The requirement on error probability, which tells us how reliable the transmission is.
- The available bandwidth (or spectral) efficiency, which measures the efficiency in bandwidth expenditure.
- The signal-to-noise ratio (SNR) necessary to achieve the required QoS. This gives us a measure on how efficiently the C/M scheme makes use of the available power.

- Implementation complexity, which is a measure of the cost of the equipment.

3.1.1 BICM: How to Obtain Top Performance Using a Single Code and Various Modulations

Forward Error Correcting codes for satellite applications must combine power efficiency and low BER floor with flexibility and simplicity to allow for high-speed implementation. The existence of practical, simple, and powerful such coding designs for binary modulations has been settled with the advent of turbo codes [2] and the recent re-discovery of Low-Density Parity-Check (LDPC) codes [17]. In parallel, the field of channel coding for non-binary modulations has evolved significantly in the latest years. Starting with Ungerboeck's work on Trellis-Coded Modulation (TCM) [11], the approach had been to consider channel code and modulation as a single entity, to be jointly designed and demodulated/decoded.

Schemes have been published in the literature, where turbo codes are successfully merged with TCM [16]. Nevertheless, the elegance and simplicity of Ungerboeck's original approach gets somewhat lost in a series of ad-hoc adaptations; in addition the turbo-code should be jointly designed with a given modulation, a solution impractical for system supporting several constellations. A new pragmatic paradigm has been crystallized under the name of Bit-Interleaved Coded Modulation (BICM) [10], where extremely good results are obtained with a standard non-optimized, code. An additional advantage of BICM is its inherent flexibility, as a single mother code can be used for several modulations, an appealing feature for broadband satellite communication systems where a large set of spectral efficiencies is needed.

3.1.2 Choice of Modulation Scheme

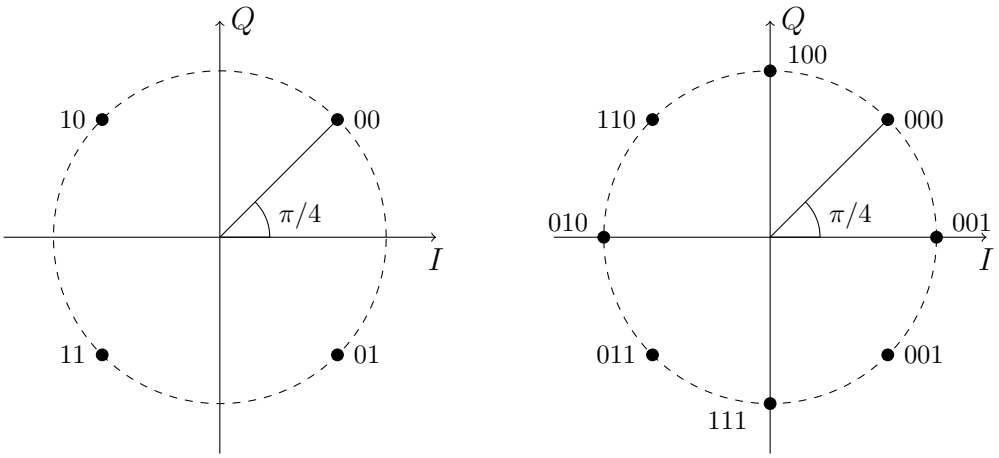
Some key issues, such as usage of increasingly high frequency bands (e.g., from X, Ku, up to Ka and Q) and large deployable spacecraft antennas providing high gain, together with availability of high on-board power attainable by large platforms currently subject of studies, have moved the focus from good-old modulation schemes, such as QPSK, to higher order modulation schemes, better providing the high data rate required from up-to-date applications such as internet via satellite, or DVB DNSG (Digital Satellite News Gathering).

Digital transmissions via satellite are affected by power and bandwidth limitations. Satellites, in fact, were originally born power limited and this limitation is still the major constraint in satellite communications. Therefore DVB-S2 provides for many transmission modes (FEC coding and modulations), giving different tradeoffs between power and spectrum efficiency. Code rates of 1/4, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5, 5/6, 8/9 and 9/10 are available depending on the selected modulation

and the system requirements. Coding rates 1/4, 1/3 and 2/5 have been introduced to operate, in combination with QPSK, under exceptionally poor link conditions, where the signal level is below the noise level. Computer simulations demonstrated the superiority of such modes over BPSK modulation combined with code rates 1/2, 2/3 and 4/5. The introduction of two FEC code block length (64800 and 16200) was dictated by two opposite needs: the C/N performance improves for long block lengths, but the end-to-end modem latency increases as well. Therefore for applications not critical for delays (such as for example broadcasting) the long frames are the best solution, while for interactive applications a shorter frame may be more efficient when a short information packet has to be forwarded immediately by the transmitting station. Four modulation modes can be selected for the transmitted payload Modulation formats provided by DVB-S2:

QPSK and 8-PSK which are typically employed in broadcast application since they have a constant envelope and therefore can be used for in nonlinear satellite transponders near to saturation operating regime;

16-APSK and 32-APSK schemes can also be used for broadcasting, but these require an higher level of available C/N ratio and, furthermore, the adoption of pre-distortion technique.

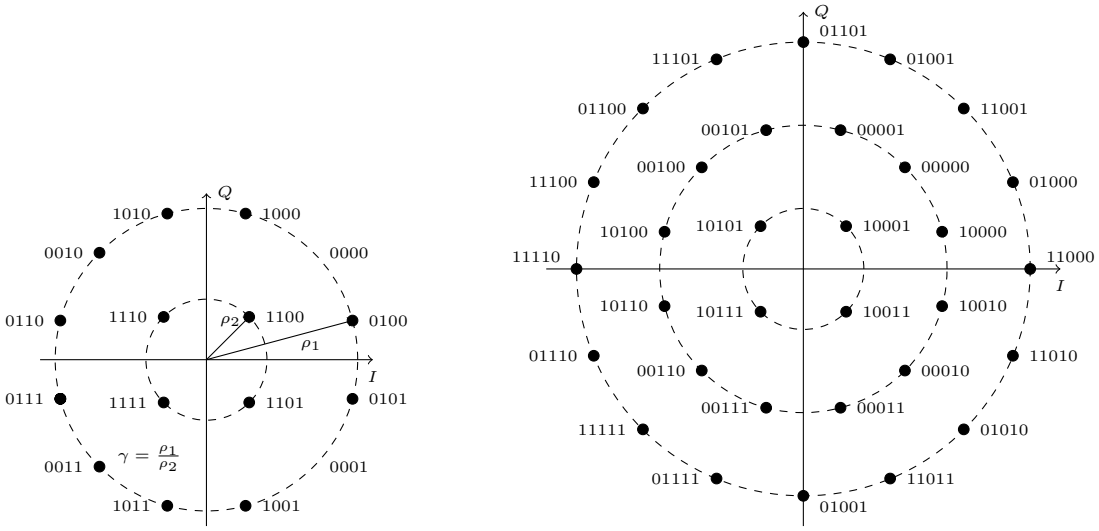


(a) QPSK (b) 8-PSK

Figure 3.1: QPSK and 8-PSK DVB-S2 constellations with normalized radius $\rho = 1$

3.1.3 The Ultimate Bound to Compare Modem Performance

As said by Shannon in 1948, year of publication of his paper [3], "the fundamental problem of communication is that of reproducing either exactly or approximately



(a) 16-APSK (b) 32-APSK

Figure 3.2: APSK constellations with normalized mean energy

a message selected at the another point". Shannon in his work set up the basis of the Information Theory. As also pointed out by Hartley, he defined as the most natural choice the logarithmic measure of information, putting in relation reliable transmission with statistical characterizations of channels.

From an engineering point of view, one of the most important point raised up by Shannon's work is the relationship between the mean power available at the transmitters and the communications reliability. One of most important aims in satellite applications is often finding the best trade-offs between the above quantities. This is prevalently due to limited power resources on-board which have to be exploited in the most efficient way. For example, wastes of power would provoke the decreasing not only of the satellite mean lifetime, but also the communication throughput.

By using Shannon theorem upon channel capacity, an important limit on communications performance is achievable. From his theorem in [3], the capacity of a discrete-time additive white Gaussian noise (AWGN) channel is given by

$$C = \frac{1}{2}\log_2(1 + \frac{S}{N})\,\text{bits/transmission} \tag{3.1}$$

When dealing with a continuous-time, bandlimited, additive white Gaussian noise channel with noise power spectral density $\frac{N_0}{2}$, input power constraint P, and bandwidth W, one can sample at the Nyquist rate[1] and obtain a discrete time channel.

[1]Sampling at Nyquist rate simply means to eliminate inter-symbol interference (ISI) while making channel discrete and memoryless. In other words, each symbol to be transmitted over the channel is made independent (obviously at the output) from the others.

The power per sample will be P and the noise power per sample will be WN_0. Substituting these results in (3.1), we obtain

$$C = \frac{1}{2}\log_2\left(1 + \frac{P}{N_0 W}\right) \text{ bits/transmission} \tag{3.2}$$

If we multiply this result by the number of transmissions per second, which is $2W$, we obtain the channel capacity in bps

$$C = W\log_2\left(1 + \frac{P}{N_0 W}\right) \text{ bps} \tag{3.3}$$

From this result, the basic factors that determine the channel capacity are the channel bandwidth W, the noise power spectrum N_0, and the signal power P. There exists a tradeoff between P and W in the sense that one can compensate for the other.

Increasing the input signal power obviously increases the channel capacity, because when one has more power to spend, one can choose a larger number of input levels that are far apart, and therefore more information bits per transmission are possible. However, the increase in capacity as a function of power is logarithmic and slow. This is because if one is transmitting with a certain level of immunity against noise and wants to increase the number of input levels, one has to introduce new levels with amplitudes higher than the existing levels, and this requires a lot of power. This fact notwithstanding, the capacity of the channel can be increased to any value by increasing the input power.

The effect of the channel bandwidth, however, is quite different. Increasing W has two contrasting effects. On one hand, on a higher bandwidth channel one can transmit more samples per second and therefore increase the transmission rate. On the other hand, a higher bandwidth means higher input noise to the receiver and this degrades its performance. Letting the bandwidth W tend to infinity, one can observe that, contrary to the power case, capacity can not be increased to any desired value by increasing the bandwidth alone.

In any practical communication system we must have $R < C$. If an AWGN channel is employed, we have

$$R < W\log_2\left(1 + \frac{P}{N_0 W}\right) \tag{3.4}$$

By dividing both sides by W and defining $\eta = \frac{R}{W}$, the spectral efficiency measured in $\frac{\text{bit}}{\text{Hz}}$, we obtain

$$\eta < \log_2\left(1 + \frac{P}{N_0 W}\right) \tag{3.5}$$

If E_b is the energy per bit, then $E_\mathrm{b} = \frac{P}{R}$. By substituting in the previous relation, we obtain[2]

$$\eta < \log_2\left(1 + \eta\frac{E_\mathrm{b}}{N_0}\right) \tag{3.6}$$

This relation is plotted in Figure 3.3. The curve defined by

$$\frac{E_\mathrm{b}}{N_0} = \log_{10}\left(\frac{2^\eta - 1}{\eta}\right) \tag{3.7}$$

divides the plane in two regions. In one region (above the curve) reliable communication is possible, and in the other region (below the curve) reliable communication is not possible. The performance of any communication system can be denoted by a point in this plane and the closer the point is to this curve, the closer is the performance of the system to an optimal system. From this curve it is seen that

$$\frac{E_\mathrm{b}}{N_0} = \ln 2 = 0{,}693 \sim -1{,}592\,\mathrm{dB} \tag{3.8}$$

is an absolute minimum for reliable communication. In other words, for reliable communications, we must have

$$\frac{E_\mathrm{b}}{N_0} > 0{,}693 \tag{3.9}$$

In Figure 3.3, when $\eta \ll 1$, we are dealing with a case where bandwidth is large and the main concern is limitation on power. This case is usually referred to as the *power limited case*. Signaling schemes with high dimensionality, such as orthogonal, biorthogonal, and simplex, are frequently used in these cases. The case, where $\eta \gg 1$ happens when the bandwidth of the channel is small, and therefore is referred to as the *bandwidth limited case*. Low-dimensional signaling schemes with crowded constellations are implemented in this case.

Plotting on the spectral efficiency–average power plane performance of several modulation schemes, we could observe that the SNR penalty (referred to Shannon limit) decreases with increasing of the modulation order M. In other words, at the expense of an higher complexity, performance very close to Shannon limit can be achieved. However, in this manner, the system would operate at spectral efficiencies excessively small or big if compared to those commonly required.

As we shall see in details in Section 3.4, Figure 3.14 shows performance for each modulation and coding (C/M) scheme adopted by DVB-S2 modulator with respect

[2]If we define W as the the equivalent noise bandwidth on the Nyquist shaping filter, which is typically chosen in communications to eliminate ISI contribution, we obtain independently from any roll-off factor that $W = R_\mathrm{b}$. Note that, in fact, for the specific symmetry of Nyquist pulse the equivalent noise bandwidth on this pulse does not change with roll-off factor variations. Hence the reason of simplification $\frac{E_\mathrm{b}R_\mathrm{b}}{N_0W} = \frac{E_\mathrm{b}}{N_0}$ made in this relation.

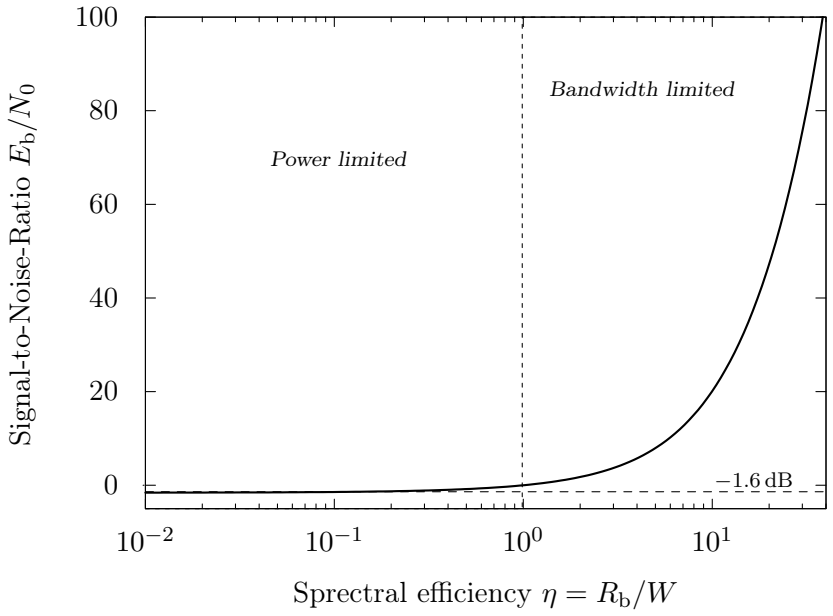


Figure 3.3: Shannon Bound: On the horizontal axis is reported spectral efficiency η, on the vertical axis Signal-to-Noise Ratio measured in dB; Dotted line at $-1.6\,\mathrm{dB}$ represents the asymptotic value for $\eta \simeq 0$; Heavy line is the Shannon curve above which every point must stays

to the calculated Shannon limit. Thanks to an optimized choice on C/M schemes, the DVB-S2 modulator provides a wide range of spectral efficiencies (see Table 3.7) while ensuring performance very close to the Shannon limit.

3.1.4 Adaptive Coding Modulation

Adaptive Coding Modulation (ACM) technique allows different modulation formats and error protection levels (i.e. coding rates) to be used and changed on a frame-by-frame basis within the transmitted data stream. By means of a return channel, informing the transmitter of the actual receiving condition, the transmission parameters may be optimized for each individual user, dependant on link conditions. Today ACM is considered as a powerful tool to further increase system capacity, allowing for better utilization of transponder resources.

Since it well-known that the higher the frequency bands used, the less the link performance tends to remain location-independent and stationary (i.e. stable space wise), countermeasures to counteract the above link variations are required. One can adopt static or dynamic solution: adopting static countermeasures against propagation effects simply means to dimension link parameters (e.g. antenna diameter,

power) on worst case basis (link budgets usually contain an item called 'propagation margin at nn % of time'); on the other hand, the most common form of dynamic countermeasure is power control, a technique to manage both propagation phenomena. Despite the beneficial power control effect on system efficiency, when for instance a user up-link operates in clear weather or in light of traffic conditions, the potentiality of that link is not fully exploited. In other words power control helps, but does not solve to the problem of system parameters over-sizing.

Current point-to-point multi-beam satellite systems based on the DVB-S standard are designed for link closure in the worst-case propagation and location conditions. The old DVB-S standard, envisaged for broadcasting applications, considers a fixed coding rate and modulation format, which are selected according to the assumed coverage and availability requirements. This approach implies the necessity of high margins in the majority of the cases, when interference and propagation conditions allow for higher Signal to Interference plus Noise Ratio (SNIR). Typical Ku-band broadcasting links are designed with a clear-sky margin of 4 dB to 6 dB and a service availability target of about 99% of the worst month (or 99,6% of the average year). Since the rain attenuation curves are very steep in the region 99% to 99,9% of the time, many dBs of the transmitted satellite power are useful, in a given receiving location, for only some ten minutes per year. Unfortunately, this waste of satellite power/capacity cannot be easily avoided for broadcasting services, where millions of users, spread over very large geographical areas, receive the same content at the same time.

However, this design methodology, devised for broadcasting systems, is not optimal for unicast networks. In fact, the point-to-point nature of link connections allows exploiting spatial and temporal variability of end-user channel conditions for increasing average system throughput. This is achieved by adapting coding rate and modulation format (ACM) to best match the user SNIR, thus making the received data rate location and time dependent. Fixed link margins are therefore avoided and a remarkable improvement in system capacity is obtained thanks to better utilization of system power resources.

Assuming a fixed beam power allocation, the key parameters responsible for SNIR variability within the satellite coverage are the fading attenuation, the satellite antenna gain and the antenna C/I. The fading attenuation variation is due to both geographical dependency of rain statistics and to propagation channel time variations.

One of the key factors, which can heavily affect capacity, is the granularity of physical layer schemes supported by the system. It is important to ensure a reasonably small step in E_s/N_0 and spectral efficiency between consecutive schemes in order to maximize the data rate provided to STs. In particular, it is of the highest importance for capacity purposes utilizing a sufficiently fine granularity in the SNIR range experienced in the coverage during clear sky conditions. On the

contrary, larger SNIR intervals can be assumed between the more robust modes without major capacity penalty, due to their lower occurrence probability.

The DVB-S2 ACM modulator operates at constant symbol rate, since the available transponder bandwidth is assumed constant. ACM is implemented by the DVB-S2 modulator by transmitting a TDM sequence of physical layer frames, each transporting a coded block and adopting a uniform modulation format. However, when ACM is implemented, coding scheme and modulation format may change frame-by-frame. Therefore service continuity is achieved, during rain fades, by reducing user bits while increasing at the same time FEC redundancy and/or modulation ruggedness.

ACM technique therefore allows to significantly increase the average system throughput and availability of satellite networks, thus making the system economically more attractive for interactive applications. The increase is mainly dependant on the adopted frequency band, target link and service area availability requirements and related system sizing options. Compared to Constant Coding and Modulation (CCM), an ACM-based system can provide:

- higher system throughput than the one supported by a CCM system, as an ACM system can take benefit from better link propagation and beam C/I conditions than the worst case link on which CCM physical layer is sized;
- higher availability (time-link and/or spatial) than the one supported by the CCM system as when deeper fading occurs the ACM system can use a more robust modulation and coding scheme. This is obtained through a very small reduction of the total system throughput as the number of users affected by these deep fade events is very limited. The highest link availability and service area supported by the system depends on the lowest modulation and coding scheme supported by the system;
- more optimization dimensions in the system design to cope with more pushed frequency reuse and more complex satellite antennas.

3.2 ACM Modem System Description

DVB-S2 ACM Modem system is defined as the functional block of equipment performing the adaptation of the baseband digital signals, from the output of a single (or multiple) MPEG transport stream multiplexer(s), or from the output of a single (or multiple) generic data source(s), to the satellite channel characteristics. Data services may be transported in Transport Stream format (e.g. using Multi-protocol Encapsulation[3]) or Generic Stream format.

[3]It is a means to allow more than a packet format (e.g., IP) over MPEG frames.

If the received signal is above the $C/N + I$ threshold, the Forward Error Correction (FEC) technique adopted in the System is designed to provide a 'Quasi Error Free' (QEF) quality target. The definition of QEF adopted for DVB-S2 is "less than one uncorrected error-event per transmission hour at the level of a 5 Mbit/s single TV service decoder", approximately corresponding to a Transport Stream Packet Error Ratio PER $< 10^{-7}$ before de-multiplexer.

3.2.1 Architecture and Sub-Blocks Specifications

According to Figure 3.4, the DVB-S2 System is composed of a sequence of functional blocks as described below.

Mode Adaptation is application dependent. It provides input stream interfacing, Input Stream Synchronization (optional), null-packet deletion (for ACM and Transport Stream input format only), CRC-8 coding for error detection at packet level in the receiver (for packetized input streams only), merging of input streams (for Multiple Input Stream modes only) and slicing into DATA FIELDs. For Constant Coding and Modulation (CCM) and single input Transport Stream, Mode Adaptation consists of a "transparent" DVB-ASI (or DVB-parallel) to logical-bit conversion and CRC-8 coding.

A Base-Band Header is appended in front of the Data Field, to notify the receiver of the input stream format and Mode Adaptation type. To be noted that the MPEG multiplex transport packets may be asynchronously mapped to the Base-Band Frames.

Stream Adaptation is applied to provide padding to complete a Base-Band Frame and Base-Band Scrambling.

Forward Error Correction (FEC) Encoding is carried out by the concatenation of BCH outer codes and LDPC (Low Density Parity Check) inner codes (rates 1/4, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5, 5/6, 8/9, 9/10). Depending on the application area, the FEC coded blocks have length $n_{\mathrm{ldpc}} = 64800$ bits or 16 200 bits. When VCM and ACM is used, FEC and modulation mode may be changed in different frames, but remains constant within a frame. Bit interleaving is applied to FEC coded bits for 8PSK, 16APSK and 32APSK.

Mapping into QPSK, 8PSK, 16APSK and 32APSK constellations is applied, depending on the application area. Gray mapping of constellations is used for QPSK and 8PSK.

Physical Layer Framing Physical layer framing should be applied, synchronous with the FEC frames, to provide Dummy PLFRAME insertion, Physical Layer

(PL) Signalling, pilot symbols insertion (optional) and Physical Layer Scrambling for energy dispersal. Dummy PLFRAMEs are transmitted when no useful data is ready to be sent on the channel. The System provides a regular physical layer framing structure, based on SLOTs of $M = 90$ modulated symbols, allowing reliable receiver synchronization on the FEC block structure. A slot is devoted to physical layer signalling, including Start-of-Frame delimitation and transmission mode definition. This mechanism is suitable also for VCM and ACM demodulator setting. Carrier recovery in the receivers may be facilitated by the introduction of a regular raster of pilot symbols ($P = 36$ pilot symbols every 16 SLOTs of 90 symbols), while a pilot-less transmission mode is also available, offering an additional 2,4% useful capacity.

Base-Band Filtering and Quadrature Modulation is applied to shape the signal spectrum (squared-root raised cosine, roll-off factors 0,35 or 0,25 or 0,20) and to generate the RF signal.

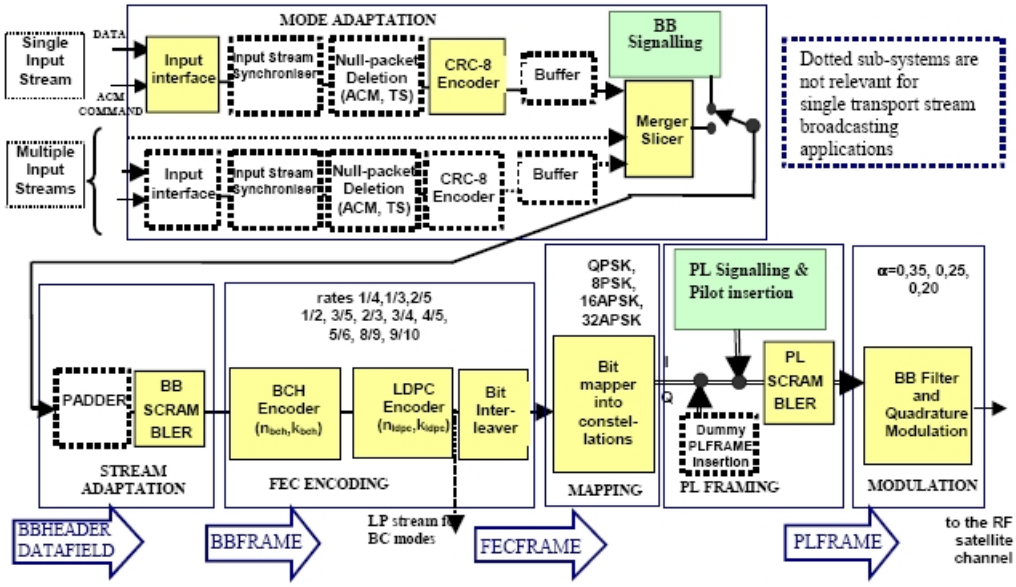


Figure 3.4: Functional block diagram of DVB-S2 ACM modem architecture

3.2.2 Sub-Blocks Description

The following subsystems description is organized according to the functional block diagram in Figure 3.4.

Mode Adaptation

DVB-S2 modem architecture deals with various input sequences:

- Single or multiple Transport Streams (TS), which are characterized by User Packets (UPs) of constant length equal to UPL = 188 bytes (one MPEG packet) whose first byte is a sync-byte (47_{HEX});
- Single or multiple Generic Streams (GS), which are characterized by continuous bit streams or streams of constant length user packets.

The first input bit is interpreted as the most significant bit (MSB). ACM command signalling allows setting of the transmission parameters to be adopted by DVB-S2 modulator, for a specific portion of input data.

Since the DVB-S2 modulator may produce variable transmission delay on the user information, the Input Stream Synchronizer provides a means to guarantee a constant bit rate (CBR) and constant end-to-end transmission delay.

The identification and erasure of MPEG null-packets allow to reduce the information rate and increase the error protection in the modulator. This process is carried out in a way that the removed null-packets can be re-inserted in the receiver in the exact place where they originally were. Specifications on this process are available in dedicated annex in [6].

A systematic Cyclic Redundancy Check (CRC) encoding is provided so that the receiver can detect the presence of errors in received streams. The CRC of 8 bits is inserted in the Base-Band Header (BBHEADER), which has the overall length of 80 bits. BBHEADERs contain also other signaling information such as indications on the roll-off factor of the shaping filter (Square Root Raised Cosine), the presence/absence of padding bits, Adaptive/Constant Coding Modulation, the use of identification and erasure of null-packet function, etc.

Stream Adaptation

Stream adaptation (see Figure 3.5) provides padding to complete a constant length (k_{bch} bits) BBFRAME. k_{bch} depends on the FEC rate, as reported in Table 3.5. Padding may be applied in circumstances when the user data available for transmission are not sufficient to completely fill a BBFRAME, or when an integer number of UPs has to be allocated in a BBFRAME. The input stream consists of a BBHEADER followed by a DATA FIELD. The output stream will be a BBFRAME.

PADDING $k_{\mathrm{bch}} - \mathrm{DFL} - 80$ zero bits are appended after the DATA FIELD so that the resulting BBFRAME have a constant length of k_{bch} bits. For Broadcast Service applications, $\mathrm{DFL} = k_{\mathrm{bch}} - 80$, therefore no padding must be applied.

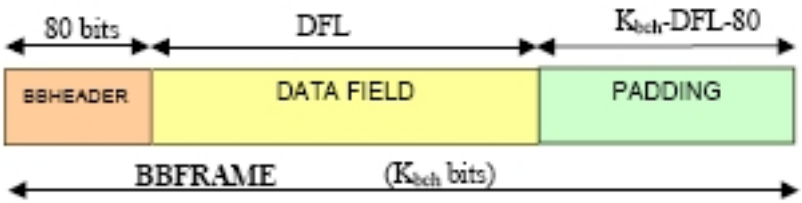


Figure 4: BBFRAME format at the output of the STREAM ADAPTER

Figure 3.5: BBFRAME format at the output of the STREAM ADAPTER

SCRAMBLING The complete BBFRAME should be randomized and the randomization sequence should be synchronous with the BBFRAME, starting from the MSB and ending after k_{bch} bits. Scrambling sequences are typically generated by feed-back shift registers. There are two main reasons why scrambling is used:

- It facilitates the work of a timing recovery circuit, an automatic gain control and other adaptive circuits of the receiver (eliminating long sequences consisting of 0 or 1 only).
- It eliminates the dependence of a signal's power spectrum upon the actual transmitted data, making it more dispersed to meet maximum power spectral density requirements (because if the power is concentrated in a narrow frequency band, it can interfere[4] with adjacent channels due to the cross modulation and the intermodulation caused by non-linearities of the receiving tract).

FEC Encoding

This subsystem performs Outer Coding (BCH), Inner Coding (LDPC) and Bit Interleaving. The input stream is composed of BBFRAMEs and the output stream of FECFRAMEs.

Each BBFRAME (k_{bch} bits) is processed from the MSB to LSB by the FEC coding subsystem to generate a FECFRAME (n_{ldpc} bits). The parity check bits (BCHFEC) of the systematic BCH outer code must be appended after the BBFRAME and the parity check bits (LDPCFEC) of the inner LDPC encoder must be appended after the BCHFEC field, as shown in Figure 3.6.

For 8PSK, 16APSK, and 32APSK modulation formats, the output of the LDPC encoder is bit interleaved using a block interleaver (see Figure 3.7 and Figure 3.8).

[4]Standardization committees such as ETSI (European Telecommunications Standard Institute) have imposed various rules as to interferences.

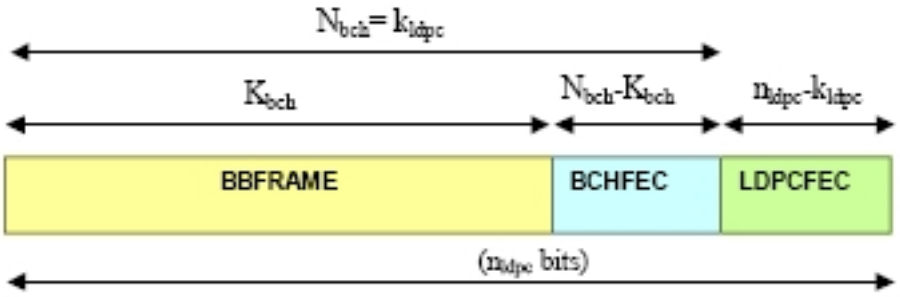


Figure 3.6: Format of data before Bit Interleaving

Data is serially written into the interleaver column-wise, and serially read out row-wise (the MSB of BBHEADER is read out first, except 8PSK rate 3/5 case where MSB of BBHEADER is read out third) as shown in the figures.

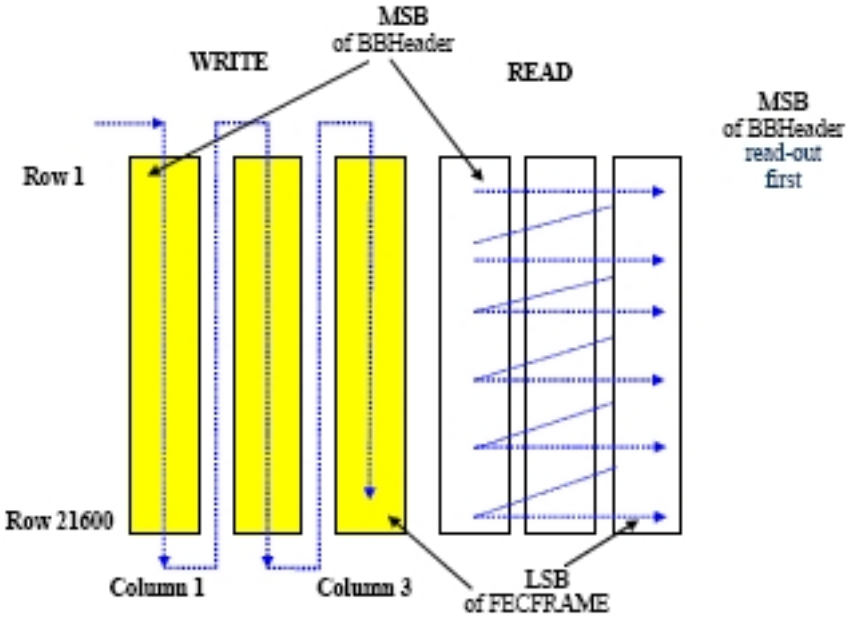


Figure 3.7: Bit Interleaving scheme for 8PSK (for each rate except from 3/5) and normal FECFRAME length

Bit Mapping

Each FECFRAME (which is a sequence of 64800 bits for normal FECFRAME, or 16200 bits for short FECFRAME), must be serial-to-parallel converted (parallelism

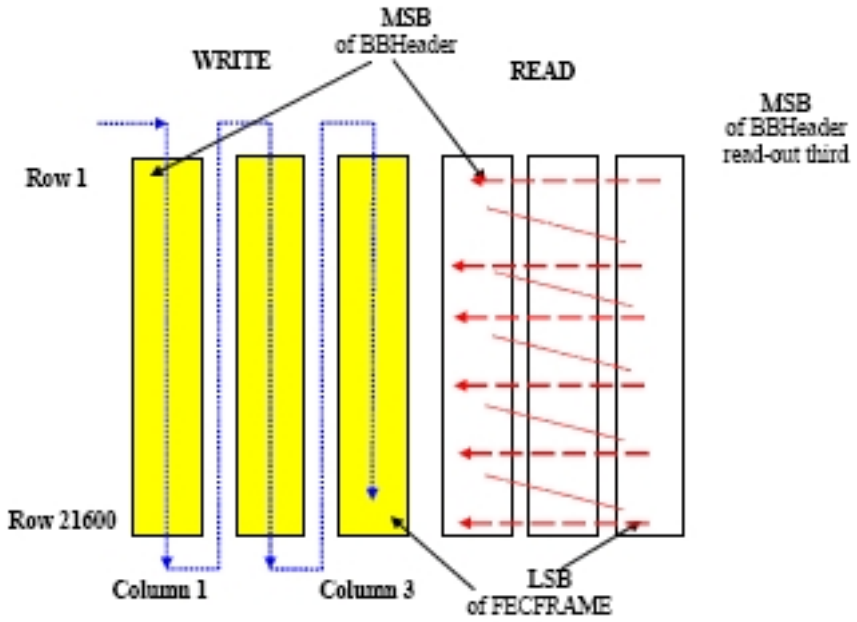


Figure 3.8: Bit Interleaving scheme for 8PSK (rate 3/5) and normal FECFRAME length

Modulation	Rows ($n = 64800$)	Rows ($n = 16200$)	Columns
8PSK	21600	5400	3
16APSK	16200	4050	4
32APSK	12960	3240	5

Table 3.1: Bit Interleaver structure

level η_{MOD}= 2 for QPSK, 3 for 8PSK, 4 for 16APSK, 5 for 32APSK) in figures 9 to 12, the MSB of the FECFRAME is mapped into the MSB of the first parallel sequence. Each parallel sequence shall be mapped into constellation, generating a (I,Q) sequence of variable length depending on the selected modulation efficiency η_{MOD}.

The input sequence is a FECFRAME, the output sequence is a XFECFRAME (compleX FECFRAME), composed of $\frac{64800}{\eta_{\mathrm{MOD}}}$ (normal XFECFRAME) or $\frac{16200}{\eta_{\mathrm{MOD}}}$ (short XFECFRAME) modulation symbols. Each modulation symbol is a complex vector in the format (I,Q) (I being the in-phase component and Q the quadrature component) or in the equivalent format $\rho \exp(\mathrm{j}\phi)$($\rho$ being the modulus of the vector and ϕ being its phase).

For QPSK, the System employs conventional Gray-coded QPSK modulation with absolute mapping (no differential coding). Bit mapping into the QPSK constellation

is illustrated in Figure 3.1. The normalized average energy per symbol is equal to $\rho^2 = 1$. Two FECFRAME bits are mapped to a QPSK symbol, i.e. bits $2i$ and $2i + 1$ determines the i-th QPSK symbol, where $i = 0, 1, 2, \ldots, (N/2) - 1$ and N is the coded LDPC block size.

For 8PSK, the System employs conventional Gray-coded 8PSK modulation with absolute mapping (no differential coding). Bit mapping into the 8PSK constellation is shown in Figure 3.1. The normalized average energy per symbol is equal to $\rho^2 = 1$. For all the rates excluding 3/5, bits $3i, 3i + 1, 3i + 2$ of the interleaver output determine the i^{th} 8PSK symbol where $i = 0, 1, 2, \ldots, (N/3) - 1$and N is the coded LDPC block size. For rate 3/5 bits $3i + 2, 3i + 1, 3i$ of the interleaver output determine the i-th 8PSK symbol where $I = 0, 1, 2, \ldots, (N/3) - 1$ and N is the coded LDPC block size.

The 16APSK modulation constellation (see figure Figure 3.1) is composed of two concentric rings of uniformly spaced 4 and 12 PSK points, respectively in the inner ring of radius ρ_1 and outer ring of radius ρ_2. The ratio of the outer circle radius to the inner circle radius ($\gamma = \rho_2/\rho_1$) should be comply with Table 3.3. If $4{\rho_1}^2 + 12{\rho_2}^2 = 16$ the average signal energy becomes 1. Bits $4i$,$4i+1$,$4i+2$ and $4i+3$ of the interleaver output determine the i-th 16APSK symbol, where $i = 0, 1, 2, \ldots, (N/4) - 1$ and N is the coded LDPC block size.

The 32APSK modulation constellation (see Figure 3.1) is composed of three concentric rings of uniformly spaced 4, 12 and 16 PSK points, respectively in the inner ring of radius ρ_1, the intermediate ring of radius ρ_2 and the outer ring or radius ρ_3. Table 3.2 defines the values of $\gamma_1 = \rho_2/\rho_1$ and $\gamma_2 = \rho_3/\rho_1$. If $4{\rho_1}^2 + 12{\rho_2}^2 + 16{\rho_3}^2 = 32$ the average signal energy becomes equal to 1. Bits $5i, 5i+1, 5i+2, 5i+3$ and $5i + 4$ of the interleaver output determine the i-th 32APSK symbol, where $i = 0, 1, 2, (N/5) - 1$.

Code Rate	η	γ_1	γ_2
3/4	3,74	2,84	5,27
4/5	3,99	2,72	4,87
5/6	4,15	2,64	4,64
8/9	4,43	2,54	4,33
9/10	4,49	2,53	4,3

Table 3.2: Optimum constellation radius ratios γ_1 and γ_2 (linear channel) for 32APSK

Code Rate	η	γ
2/3	2,66	3,15
3/4	2,99	2,85
4/5	3,19	2,75
5/6	3,32	2,7
8/9	3,55	2,6
9/10	3,59	2,57

Table 3.3: Optimum constellation radius ratio γ (linear channel) for 16APSK

Physical Layer (PL) Framing

This block generates, synchronously with FECFRAMEs, physical layer frames (PLFRAMEs) and deal with

- the insertion of physical headers and optional pilot symbols (their insertion causes a loss of 2,4% of capacity);
- the insertion of dummy frames to be used in absence of data ready to be immediately transmitted;
- the scrambling (or the randomization) for energy dispersal by multiplying the $(I + \mathrm{j}\, Q)$ samples by a complex radomization sequence $(C_{\mathrm{I}} + \mathrm{j}\, C_{\mathrm{Q}})$.

3.3 Inner and Outer FEC

The DVB-S2 FEC section relies on two block codes concatenation, i.e., codewords generated by BCH encoder are, in turn, encoded by LDPC encoder[5]. Thus, BCH code is usually called *outer code* and LDPC *inner code*. Outputs from FEC encoding section are formatted according to Figure 3.6. As we shall see later on, this BCH concatenation gives an extra protection [6] against unwanted error floors at high SNR[6] When the decoding has to be performed, the above sequence of encoding steps must be, of course, inverted: in other words, LDPC starts to decode prior

[5]Note that each frame (either short or long) is processed from most significant to least significant bit, either by BCH or LDPC.

[6]The *iterative decoding* achieves very close to Shannon limit performance at low/medium SNR. On the contrary, its performance may be significantly worse at high SNR. In fact, their free distance (the minimum Hamming distance between different codewords, i.e., the minimum Hamming weight of a nonzero codeword) can be low. This causes BER curve to flatten following the error floor tied to d_{free}, after the waterfall region at low SNR. Performance of any binary code at high SNR can be well approximated by the expression of union bound, truncated to the contribution of free distance

to BCH. As just said, error floor phenomena at high signal-to-noise ratio typically affect the decoding performance, and therefore the BCH extra protection against residual errors is aimed at counteracting to the performance degradation, enhancing FEC robustness at higher SNR.

In a some sense, the chain constituted, in order, by LDPC encoder, channel and LDPC decoder can be conceived as a super-channel on which BCH encoder/decoder operates. This kind of intuitive representation puts in evidence the advantage of concatenating two codes. The best choices on their kind of concatenation can give a powerful tool to enhance the performance of the overall FEC decoding, allowing to act by a outer coding on the weaknesses of the inner coding.

3.3.1 BCH

BCH code is named for Bose, Ray-Chaudhury, and Hocquenghem, who published work in 1959 and1960 which revealed a means of designing codes over $GF(2)$ with a specified design distance. BCH codes are cyclic codes and thus, as explained in Appendix B, can be specified by a generator polynomial. In present Section we give only some notations over the particular, as we shall see, structure of BCH code employed in DVB-S2 and, more generally, over its design steps.

A BCH code of length n with a specified minimum distance, or, i.e., capable of correcting (at least) t errors, can be designed as follows [15, 9]:

1. Determine the smallest m such that $GF(q^m)$ has primitive nth root (Section A.5) of unity β.

2. Select a nonnegative integer b. In most of cases, it is selected $b = 1$.

3. Write down a list of $2t$ consecutive powers of β (see Section A.3):

$$\beta^b, \beta^{b+1}, \ldots, \beta^{b+2t-1}$$

   Determine the minimal polynomial of each of these powers of β. (Because of conjugacy, frequently these minimal polynomials are not distinct.)

4. The generator polynomial $g(x)$ is least common multiple (LCM) of these minimal polynomials. The code is a $(n, n - \deg(g(x)))$ cyclic code.

(terms at higher distance are negligible at high SNR). Such expression [18] is

$$\mathrm{BER} \simeq \frac{1}{2}\frac{w_{\mathrm{free}}}{k}\mathrm{erfc}\left(\sqrt{d_{\mathrm{free}}\frac{k}{n}\frac{E_{\mathrm{b}}}{N_0}}\right)$$

Two fields are involved in the construction of the BCH codes. The small field $GF(q)$ is where the generator polynomial has its coefficients and is the field where the elements of the codeword are. The big field $GF(q^m)$ is the field where the generator polynomial has its roots. For encoding purposes it is only sufficient to have the polynomial generator of the code, whereas, for decoding, the additionally knowledge of the extension field where the generator has its roots is necessary. Two definitions will be useful later on: first, if β selected is a primitive element then the code is called *primitive*; second, when $\beta = 1$ the code is said to be a code in *narrow-sense*.

It can be proved that following the constructive procedure described above produces codes with at least the specified minimum distance. As shown in Appendix B, since every codeword is a multiple of the generator polynomial, we can express the *parity check condition* as follows

$$c(\beta^i) = m(\beta^i)g(\beta^i) = m(\beta^i) \cdot 0 \tag{3.10}$$

for $i = b, b+1, \ldots, b+2t-1$. In fact these powers of β are, by design, the roots of $g(x)$. Rewriting these condition using the correspondent vector representation we get the following system of equations

$$c_0 + c_1\beta^i + c_2\beta^{i+1} + \ldots + c_{n-1}(\beta^i)^{n-1} = 0 \tag{3.11}$$

always for $i = b, b+1, \ldots, b+2t-1$. Therefore $2t$ parity check conditions can be also expressed in the matrix form

$$\begin{pmatrix} 1 & \beta^i & \beta^{2i} & \cdots & \beta^{(n-1)i} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{pmatrix} = 0 \tag{3.12}$$

Stacking the row vectors for different values of i we get the following parity check matrix

$$\begin{pmatrix} 1 & \beta^b & \beta^{2b} & \cdots & \beta^{(n-1)b} \\ 1 & \beta^{b+1} & \beta^{2(b+1)} & \cdots & \beta^{(n-1)(b+1)} \\ \vdots & & & & \\ 1 & \beta^{b+\delta-3} & \beta^{2(b+\delta-3)} & \cdots & \beta^{(n-1)(b+\delta-3)} \\ 1 & \beta^{b+\delta-2} & \beta^{2(b+\delta-2)} & \cdots & \beta^{(n-1)(b+\delta-2)} \end{pmatrix} \tag{3.13}$$

where $\delta = 2t + 1$ represent the *design distance* (Hamming metric) of the code.

It can be proved (see [15]) that the matrix $\mathbf{H}$ have at least $\delta = 2t + 1$ columns linearly dependent and, thus, the minimum distance of the code satisfies $d_{\min} \geq \delta$; that is, the code is capable of correcting at least t errors.

In DVB-S2 context, BCH outer code has been designed to avoid "unpredictable and unwanted error floors which affect iterative decodings at high C/N ratios", i.e., at low bit error rates (BER).

Although the mathematical background shown in this thesis in order to design, encoding and decoding is completely general, in the most practical cases the base fields of BCH codes is $GF(2)$, as in DVB-S2 FEC section. The polynomials which must be multiplied in order to obtain the wanted BCH code associated to a specific operating mode (or, i.e., error correction capability) are shown in Table 3.4. More specifically, multiplication of the first t of them provides a polynomial generator of the BCH code capable of correcting at least t error. Notice that this procedure is in accordance with the third step to construct generator polynomial of the BCH code.

The encoding suggestions, provided in [6], are typical of a cyclic codes in systematic form, whose specific properties and encoding steps are discussed in dedicated Appendix B. It is worth understanding that both systematic and non-systematic codewords are always multiple of the generator polynomial of the code (or, i.e., of the ideal, as also shown in Section B.2) so that parity check condition (3.12) is still valid as well as BCH bound. Furthermore, from a direct analysis of the polynomial supplied by [6] (Normal FEC-Frame), one could observe that these specific codes (one for each selected error protection level) are *primitive* and, indeed, *narrow-sense*. The reasons are described in the following

- $g_1(x)$ is one of the possible primitive polynomial which has its roots in $GF(2^{16})$. This can be quickly proved verifying that $\alpha^{2^{16}-1} = 1$ by computer aid or, more lazily, by direct consultation of any table of primitive polynomials. For the tables of primitive polynomials, the interested reader could refer to [21]

- Hence, this BCH is a narrow-sense code. In fact, by design, it follows that b must be equal to 1.

Three different t-error correcting BCH codes ($t = 12$, $t = 10$, $t = 8$) should be applied to each BBFRAME, according to LDPC coding rates as shown in Table 3.5. Outer FEC (i.e. BCH), although there are 11 LDPC coding rates, deal with only three different error protection level: $t = 12, t = 10, t = 8$. On the one hand we have, since this BCH is primitive, that the codeword length must be equal to $2^{16} - 1$. On the other hand we have in Table 3.5 multiple codeword lengths for each t-BCH code and all of them do not correspond to the primitive length BCH should have. In general, for each code rate, a different set of polynomials to be multiplied would be expected, whereas, in DVB-S2, there is only one set of polynomials provided by [6].

Eventually, we have to conclude that DVB-S2 BCH code is *shortened*. A systematic (n, k) code can be shortened by setting a number of the information bits to zero

(i.e, zero padding). This means that a linear (n, k) code consisting of k information bits and $n-k$ check bits (or redundancy bits) can be shortened into a $(n-l, k-l)$ linear code by setting the most (or the least, depending on the reference direction) significant first l bits to zero. As also described in [15], shortening of a code does not have bad effects on its minimum distance properties and its decoding algorithms. However, shortened cyclic code might loose its cyclic property.

$g_1(x)$	$1 + x^2 + x^3 + x^5 + x^{16}$
$g_2(x)$	$1 + x + x^4 + x^5 + x^6 + x^8 + x^{16}$
$g_3(x)$	$1 + x^2 + x^3 + x^4 + x^5 + x^7 + x^8 + x^9 + x^{10} + x^{11} + x^{16}$
$g_4(x)$	$1 + x^2 + x^4 + x^6 + x^9 + x^{11} + x^{12} + x^{14} + x^{16}$
$g_5(x)$	$1 + x + x^2 + x^3 + x^5 + x^8 + x^9 + x^{10} + x^{11} + x^{12} + x^{16}$
$g_6(x)$	$1 + x^2 + x^4 + x^5 + x^7 + x^8 + x^9 + x^{10} + x^{12} + x^{13} + x^{14} + x^{15} + x^{16}$
$g_7(x)$	$1 + x^2 + x^5 + x^6 + x^8 + x^9 + x^{10} + x^{11} + x^{13} + x^{15} + x^{16}$
$g_8(x)$	$1 + x + x^2 + x^5 + x^6 + x^8 + x^9 + x^{12} + x^{13} + x^{14} + x^{16}$
$g_9(x)$	$1 + x^5 + x^7 + x^9 + x^{10} + x^{11} + x^{16}$
$g_{10}(x)$	$1 + x + x^2 + x^5 + x^7 + x^8 + x^{10} + x^{12} + x^{13} + x^{14} + x^{16}$
$g_{11}(x)$	$1 + x + x^2 + x^3 + x^5 + x^9 + x^{11} + x^{12} + x^{13} + x^{16}$
$g_{12}(x)$	$1 + x + x^5 + x^6 + x^7 + x^9 + x^{11} + x^{12} + x^{16}$

Table 3.4: BCH minimal polynomials for normal FECFRAME $n_{\mathrm{LDPC}} = 64800$

LDPC Code Rate	BCH Uncoded Block	BCH Coded Block	BCHFEC	BCH t-error Correction
1/4	16008	16200	192	12
1/3	21408	21600	192	12
2/5	25728	25920	192	12
1/2	32208	32400	192	12
3/5	38688	38880	192	12
2/3	43040	43200	160	10
3/4	48408	48600	192	12
4/5	51648	51840	192	12
5/6	53840	54000	160	10
8/9	57472	57600	128	8
9/10	58192	58320	128	8

Table 3.5: Coding parameters for normal FECFRAME $n_{\mathrm{LDPC}} = 64800$

3.3.2 LDPC

LDPC codes [17] were invented in 1960 by R. Gallager. They were largely ignored until the discovery of turbo codes [2] in 1993. Since then, LDPC codes have experienced a renaissance and are now one of the most intensely studied areas in coding.

Much of the effort in coding over the last 50 years has focused on the construction of highly structured codes with large minimum distance. The structure keeps the decoding complexity manageable, while large minimum distance is supposed to guarantee good performance. This approach, however, is not without its drawbacks. First, it seems that finding structured codes with large minimum distance turn out to be a harder problem than researchers imagined. Finding good codes in general is, in a sense, trivial: randomly chosen codes are good with high probability. More generally, one can easily construct good codes provided one admits sufficient *description complexity* into the definition of the code. This conflicts, however, with the goal of finding highly structured codes that have simple decodings. Second, close to capacity, minimum distance is only a poor surrogate for the performance measure of real interest. Iterative coding systems take an entirely different approach. The basic idea is the following. Codes are constructed so that the relationship between their bits (the structure of their redundancy) is *locally* simple, admitting simple *local decoding*. The local descriptions of the codes are interconnected in a complex (e.g., random-like) manner, introducing long-range relationships between the bits. Relatively high global description complexity is thereby introduced in the interconnection between the simple local structures. Iterative decoding proceeds by performing the simple local decodings and then exchanging the results, passing messages between locales across the 'complex' interconnection. The locales repeat their simple decodings, taking into account the new information provided to them from other locales. Usually, one uses a graph to represent this process. *Locales* are nodes in the graph, and interconnections are represented as edges. Thus, description complexity is introduced without adding computational complexity per se, but rather as wiring or routing complexity.

In addition to the classical representation of block codes (i.e. by $\mathbf{H}$ matrix), another useful and common way of representing an LDPC codes is through a graphical representation called a *Tanner graph* [17, 15, 5]. Tanner graphs of LDPC codes are bipartite graphs with variable nodes on one side and constraint nodes on the other. Each variable node corresponds to a bit, and each constraint node corresponds to one parity-check constraint on the bits defining a codeword. A parity-check constraint applies to a certain subset of the codeword bits, those that participate in the constraint. The parity-check is satisfied if the XOR of the participating bits is 0 or, equivalently, the modulo 2 sum of the participating bits is 0. Edges in the graph attach variable nodes to constraint nodes indicating that the bit associated with the variable node participates in the parity-check constraint associated with

the constraint node. A bit sequence associated with the variable nodes is a codeword if and only if all of the parity-checks are satisfied.

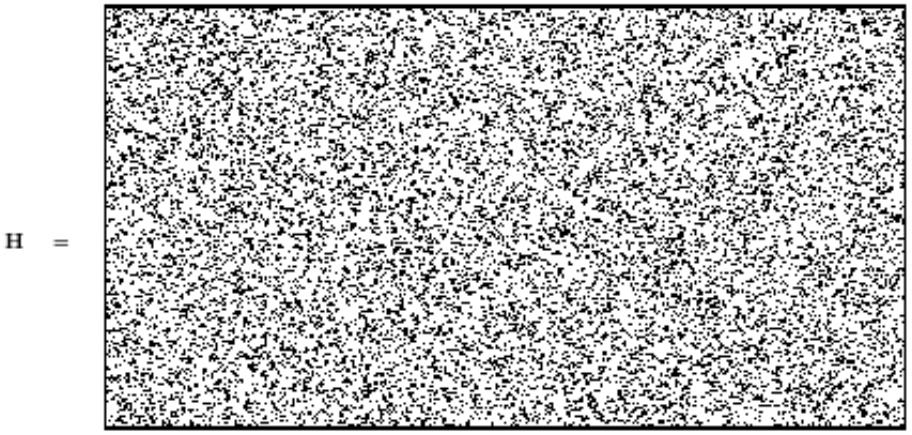


Figure 3.9: A low-density parity-check matrix with $N = 20000$ columns of weight $j = 3$ and $M = 10000$ rows of weight $k = 6$ (reference [5]).

The Tanner graph representation of the LDPC code closely mirrors the more standard parity-check matrix representation of a code. In this latter description the code is represented as the set of all binary solutions $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ to a simple linear algebraic equation (parity check equations [9, 21, 15, 5]) $\mathbf{H}\mathbf{x}^{\mathrm{T}} = \mathbf{0}^{\mathrm{T}}$. The elements of the parity-check matrix are 0s and 1s, and all arithmetic is modulo 2, that is, multiplication of $\mathbf{x}$ by a row of $\mathbf{H}$ means taking the XOR of the bits in $\mathbf{x}$ corresponding to the 1s in the row of $\mathbf{H}$. The connection between the parity-check matrix representation and the Tanner graph is straightforward, and is illustrated in Figure 3.10 by means of an example. The elements of $\mathbf{x}$ are in one-to-one correspondence with the variable nodes in the Tanner graph. Thus, the variable nodes correspond to the columns of $\mathbf{H}$. The parity checks on $\mathbf{x}$ are in one-to-one correspondence with the constraint nodes in the Tanner graph. Thus, the constraint nodes corresponds to the rows of $\mathbf{H}$. The edges in the Tanner graph correspond to the 1s in $\mathbf{H}$, that is, the entry in the i-th row and j-th column of $\mathbf{H}$ is a 1 if and only if the i-th constraint node is connected to the j-th variable node in the Tanner graph.

The Tanner graph captures the dependency structure of the various bits. The iterative decoding algorithms we will discuss work directly on this bipartite graph. Iterative algorithms for LDPC codes are message-passing and flipping algorithms, both introduced by Gallager [17].

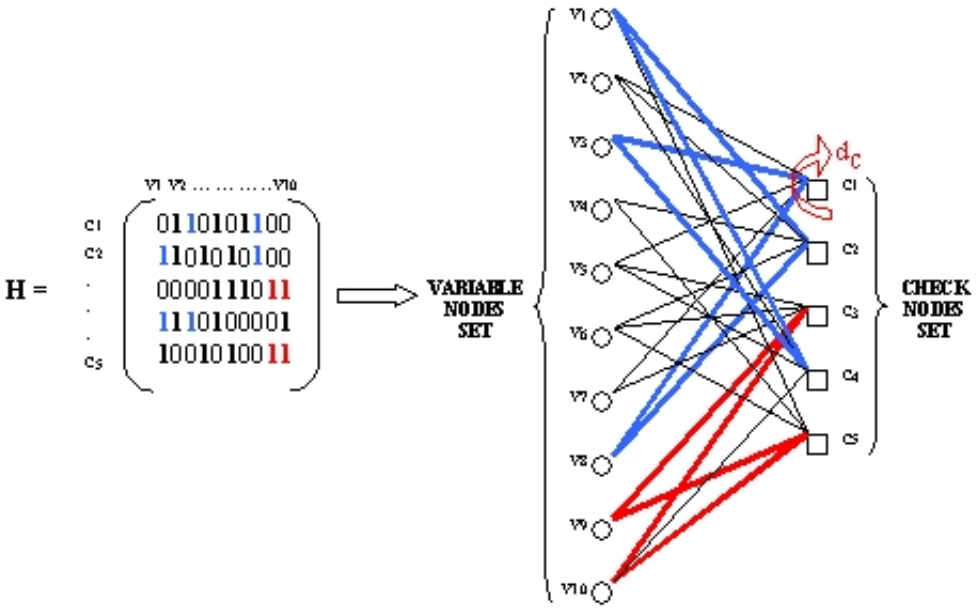


Figure 3.10: A parity check matrix **H** and the corresponding Tanner graph. To illustrate the relationship more clearly, some edge are shown in red or blue.

Iterative Decodings and Message-Passing

In *message-passing* decoders, messages are exchanged along the edges of the graph, and computations are performed at the nodes, as shown in Figure 3.11. Each message represents an estimate of the bit associated with the edge carrying the message. These decoders can be understood by focusing on one bit as follows. Suppose the bits of an LDPC codeword are transmitted over a communications channel and, during transmission, some of them are corrupted so that a 1 becomes a 0 or vice versa. Each bit node in the decoder gets to see the bit that arrived at the receiver corresponding to the one that was transmitted from the equivalent node at the transmitter. Imagine that the node would like to know if that bit is in error or not and therefore asks all of its neighboring check nodes what they think the bitŠs value should be. Each neighboring check node then asks their other neighbors what their bit values are and sends back to the original bit node the modulo 2 sum of those values. The bit node now has several opinions as to the bitŠs correct value and must somehow reconcile these opinions; it could, for example, take a majority vote. The above procedure constitutes one round or iteration of message passing.

In order to improve performance, a further such round can be performed. In more detail, assume that those bit nodes that send their observed bits to the checks to be summed first ask their check node neighbors what they think is their correct

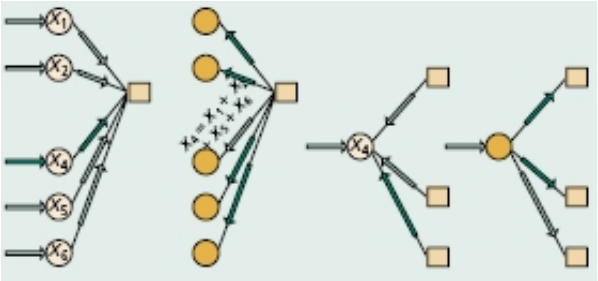


Figure 3.11: The principle of a message-passing decoder. Messages represent an estimate of the bit associated with the edge carrying the message. Nodes query neighboring nodes to collect opinions, process this information, and forward their current estimate to their neighbors.

bit value. Those check nodes could then query their other variable node neighbors and forwarded their modulo 2 sum as an opinion. With more information now available, the bit nodes would have a better chance of communicating the correct value to the check nodes, and the opinions returned to the original node would therefore have a better chance of being correct. This gathering of opinions could obviously be extended through multiple iterations; typically, many iteration rounds are performed.

In actual decoding all nodes decode concurrently. Each node gathers opinions from all its neighbors and forwards to each neighbor an opinion formed by combining the opinions of the other neighbors. This is the source of the term message passing. The process continues until either a set of bits is found that satisfies all checks or time runs out. The message passing may proceed asynchronously. Note that with LDPC codes, convergence to a codeword is easy to detect since one need only verify that the parity checks are satisfied.

The processing required for message-passing decoding LDPC codes is highly parallelizable and flexible. Each message-passing operation performed at a node depends on other nodes only through the messages that arrive at that node. Moreover, message updating need not be synchronized. Consequently, there is great freedom to distribute in time and space the computation required to decode LDPC codes. Turbo codes are also decoded using belief propagation, but their structure does not admit the same level of atomization.

There is a second distinct class of decoding algorithms that is often of interest for very high speed applications, such as optical networking. This class of algorithms is known as flipping algorithms. Bit flipping usually operates on hard decisions: the information exchanged between neighboring nodes in each iteration is a single bit. The basic idea of flipping is that each bit, corresponding to a variable node assumes a value, either 0 or 1, and, at certain times, decides whether to flip itself

(i.e., change its value from a 1 to a 0 or vice versa). That decision depends on the state of the neighboring check nodes under the current values. If enough of the neighboring checks are unsatisfied, the bit is flipped. The notion of 'enough' may be time-varying, and it may also depend on soft information available about the bit. Thus, under flipping, variable nodes inform their neighboring check nodes of their current value, and the check nodes then return to their neighbors the parity of their values. The underlying assumption is that bits that are wrong will tend to have more unsatisfied neighbors than bits that are correct.

Encoding Procedure for The DVB-S2 Code

The LDPC code of DVB-S2 standard has the following structure (see the Annex A in [7])

$$\mathbf{H} = \left[\begin{array}{c|c} \mathbf{H}^{\mathrm{d}} & \mathbf{H}^{\mathrm{p}} \end{array} \right] \tag{3.14}$$

where $\mathbf{H}^{\mathrm{d}}$ is an $r \times k$ matrix, and $\mathbf{H}^{\mathrm{p}}$ is a dual diagonal $r \times r$ matrix that is shown below for example ($r = 5$)

$$\mathbf{H}^{\mathrm{p}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \tag{3.15}$$

The notation in use is the same employed as far in standard description, that is, $r = n - k$ represents the number of redundancy bits, n the codeword length, and k the length of messages to be encoded. By now, it does not matter which rate LDPC has: clearly $\mathbf{H}$ size change with coding rates and so its composition.

Codewords, represented by vector $\mathbf{x}$, can be divided in a systematic part, namely $\mathbf{s}$, and a part relevant to the redundancy (parity) bits, namely $\mathbf{p}$, in a way such that vector $\mathbf{x}$ can be expressed as $\mathbf{x} = (\mathbf{s}, \mathbf{p})^{\mathrm{T}}$. The encoding is accomplished by searching the parity vector $\mathbf{p}$ which satisfies the (system) equation $\mathbf{Hx} = \mathbf{0}$. This system, as the $\mathbf{x}$, can be divided in two contributes as follows

$$\mathbf{H}^{\mathrm{d}}\mathbf{s} + \mathbf{H}^{\mathrm{p}}\mathbf{p} = \mathbf{0} \text{ or } \mathbf{H}^{\mathrm{d}}\mathbf{s} = \mathbf{H}^{\mathrm{p}}\mathbf{p} = \mathbf{v} \tag{3.16}$$

which is obviously equivalent to that given before.

So the encoding process can be carried out through two steps:

1. Determine the auxiliary vector $\mathbf{v} = \mathbf{H}^{\mathrm{d}}\mathbf{s}$. The number of XOR operations to do is $(n - k) \cdot (d_{\mathrm{c}} - 1)$, where d_{c} is the number of 1s in a row of $\mathbf{H}^{\mathrm{d}}$, which is constant in a regular LDPC.

2. Determine the r parity bits by a back substitution, using the equation $\mathbf{H}^{\mathrm{p}}\mathbf{p} = \mathbf{v}$, i.e.,

$$\begin{aligned} p_1 &= v_1 \\ p_l &= v_l + p_{l-1} \quad \text{for } l = 1, 2, \ldots, r \end{aligned} \tag{3.17}$$

   Here the number of operations (XOR) required is $r - 1$. This is actually a matrix inversion process, which can be almost costless executed because the matrix is bi-diagonal.

The encoder structure is shown in Figure 3.12. The overall number of operations to be performed is $n(1-R)(d_{\mathrm{c}}-1)$ (R, as usually indicated in the vast majority of literature, is the code rate) and thus the encoder complexity grows linearly with n.

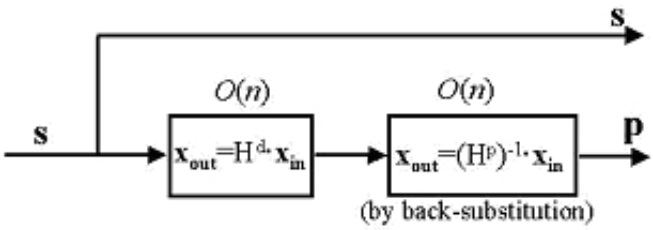


Figure 3.12: Encoding process of random-like LDPC codes

The very regular structure of $\mathbf{H}$ matrix allows to minimize the amount of memory to be allocated so as to succeed in storing the position of one in there even when using the brute force to do that. However, the full matrix, during the encoding, is recovered from compressed information, which is contained in dedicated tables for each, of course, coding rate provided.

For example, only the location of the ones in the first column of $\mathbf{H}^{\mathrm{d}}$ is described by the table (in the first table row) in the standard: the following 359 columns are obtained through a cyclic shift of the first column of a number of locations starting from the first column. In the same way, only the location of the ones in the 361st column of H is described by the table (second table row) in the standard: the following 359 columns are obtained via a cyclic shift of a number of locations starting from the 361st column. This procedure is followed up to the last (k-th) column of $\mathbf{H}^{\mathrm{d}}$.

Let us now recall the procedure indicated by ETSI in [6]. The DVB-S2 LDPC encoder systematically encodes an information block of size k_{ldpc}, $\mathbf{i} = (i_0, i_1, i_2, \ldots, i_{k_{\mathrm{ldpc}}})$ onto a codeword of size n_{ldpc}, $\mathbf{c} = (i_0, i_1, \ldots, p_0, p_1, \ldots, p_{n_{\mathrm{ldpc}}-k_{\mathrm{ldpc}}-1})$. The transmission of codewords starts in the given order from i_0 and ends with $p_{n_{\mathrm{ldpc}}-k_{\mathrm{ldpc}}-1}$. LDPC code parameters $(n_{\mathrm{ldpc}}, k_{\mathrm{ldpc}})$ are given in table These last two parameters

depend on the coding rate and the code block size, which may be either normal or short.

The procedure to calculate the parity bits for each kind of block is the following

- Set all the parity bits to zero, that is, $p_0 = p_1 = \ldots = p_{n_{\text{ldpc}}-k_{\text{ldpc}}-1} = 0$
- Accumulate the first information bit, namely i_0, at the parity address specified in the fist row of Tables B1 through B8 in Annex B [6].
- The structure of $\mathbf{H}$ matrix can be now exploited in this way: the second column, indicating over which element of the output the second information bit is accumulated, can be obtained by the first column, and so on for the other columns. Hence
  - For the next 359 information bits accumulate i_m at parity bit addresses $\{x + (r \mod 360 \times q)\} \mod (n_{\text{ldpc}} - k_{\text{ldpc}})$ where x denotes the address of the parity bit accumulator corresponding to the first bit i_0, and q is a code rate dependent constant specified in Table 7a in [6].
  - For the 361th information bit i_{360}, the addresses of parity bit accumulators are given in the second row of the tables B1 through B8 in Annex B [6]. In a similar way the addresses of the parity bit accumulators for the following 359 information bits, namely $i_m, i = 361, 362, \ldots, 719$, are obtained using the formula $\{x + (r \mod 360 \times q)\} \mod (n_{\text{ldpc}} - k_{\text{ldpc}})$ where x denotes the address of the parity bit accumulator corresponding to the information bit i_{360}, i.e., the entries in the second row of the tables B1 through B8 in Annex B [6].
- In a similar manner, for every group of 360 new information bits, a new row from tables B1 through B8 in Annex B [6] are used to find the addresses of the parity bit accumulators.

After all of the information bits are exhausted, the final parity bits are obtained as follows (this is back substitution introduced before, now executed in-place, i.e., with no use of auxiliary vector):

- Sequentially perform the following operations starting with $i = 1$

$$p_i = p_i + p_{i-1}, \; i = 0, 1, \ldots, n_{\text{ldpc}} - k_{\text{ldpc}} - 1 \tag{3.18}$$

- Final content of $p_i, i = 0, 1, \ldots, n_{\text{ldpc}} - k_{\text{ldpc}} - 1$ is equal to the parity bit p_i.

Following considerations can be done:

1. The sparse matrix $\mathbf{H}^{\mathrm{d}}$ is constituted by N_{S} sub-matrices (number of rows of tables B1 through B8) with r-rows and 360-columns, each of which is specified by a row of the tables B1 through B8; in detail, addresses of ones in the first column of each matrices is specified by a row of these tables.

2. Each column of the sub-matrices constituting $\mathbf{H}^{\mathrm{d}}$ can be obtained via a cyclic shift of the first column, in which the shift is of a number of positions (rows) equal to q. The shift is cyclic since it is reduced modulo m, the number of the rows of each sub-matrix.

3. $\mathbf{H}^{\mathrm{d}}$ has a fixed number of ones per row (d_{c}), depending on coding rate.

4. Number of ones per column in each sub-matrix is not fixed; tables B1 through B8 in fact has **col1** columns in the first **row1** rows and **col2** columns for **row2** (=N_{S}-row1) rows; this means that first **row1** sub-matrices of $\mathbf{H}^{\mathrm{d}}$ have **col1** ones per column, while **row2** sub-matrices have **col2** ones per column.

5. $\mathbf{H}^{\mathrm{p}}$ is a $r \times r$ bi-diagonal matrix.

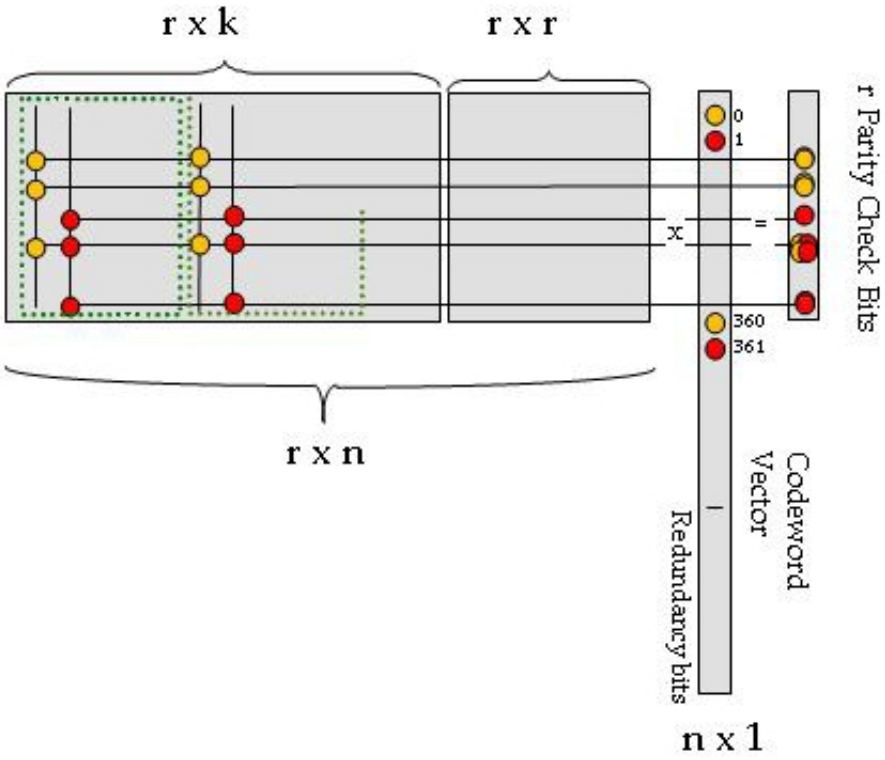


Figure 3.13: Regularity properties of the LDPC generation matrix

The code rate dependant constant q (introduced in paragraph before) allows a division of total number r of check equations in q congruence classes; each of them has 360 equations ($r = 360 \cdot q$). From this point of view and considering tables associated to LDPC coding procedure, a parity check equation can be identified by a couple of parameters R and Q: the congruence class (R_x) and the position (Q_x) in the congruence class. These parameters are obtained from following relations:

$$R_x = B_x \mod q \tag{3.19a}$$

$$Q_x = \frac{B_x}{q} \tag{3.19b}$$

where B_x is a generic element of address tables B1 trough B8, $0 < R < q - 1$ and $0 < Q < 359$. These tables are used to address systematic bits to the corresponding check equation.

All parameters of the encoding procedure are summarized in Table 3.6, where in order from left to right we have:

1. Rate: LDPC coding rate
2. $r = n - k$: number of code parity bits
3. $q = 180 \cdot (1 - R)$: code rate dependent constant
4. d_c: check nodes degree (number of bits per parity check equation)
5. $N_S = 180 - q = k/360 =$ **row1** + **row2**: number of sub-matrices $r \times 360$ of $\mathbf{H}^{NS}$
6. **row1**: number of sub-matrices with **col1** ones per column
7. **row2**: number of sub-matrices with **col2** ones per column
8. **col1**: variable nodes degree with $0 < index < 360 \cdot$ **row1**
9. **col2**: variable nodes degree with $360 \cdot$ **row1** $< index < k$
10. #Bx: total number of elements in tables B1 through B8

3.4 Modem Performance

Spectral efficiency provided by DVB-S2 ranges from about 0,5 bpsHz^{-1}, using the QPSK 1/4, to 4,5 bpsHz^{-1}, using the 32APSK 9/10; E_s/N_0 goes from −2,4 dB to 16 dB (AWGN channel and an ideal demodulation have been assumed), as also

Rate	d_c	q	N_S	k bit	r bit	**row1**	**row2**	**col1**	**col2**	#Bx
1/4	2	135	45	16200	48600	15	30	12	3	270
1/3	3	120	60	21600	43200	20	40	12	3	360
2/5	4	108	72	25920	38880	24	48	12	3	432
1/2	5	90	90	32400	32400	36	54	8	3	450
3/5	9	72	108	38880	25920	36	72	12	3	648
2/3	8	60	120	43200	21600	12	108	13	3	480
3/4	12	45	135	48600	16200	15	120	12	3	540
4/5	16	36	144	51840	12960	18	126	11	3	576
5/6	20	30	150	54000	10800	15	135	13	3	600
8/9	25	20	160	57600	7200	20	140	4	3	500
9/10	28	18	162	58320	6480	18	144	4	3	504

Table 3.6: LDPC parameters for the encoding procedure.

illustrated in Figure 3.15. This performance has been computed by computer simulations [1, 7] at a Packet Error Rate (PER) equal to 10^{-7}, corresponding about to one erroneous Transport Stream Packet per transmission hour in a 5 Mbit/s video service On an AWGN channel[7], DVB-S2 gives an increase of transmission capacity (about 20-35%) compared to DVB-S and DVB-DNSG under the same transmission conditions.

The DVB-S2 system may be used in 'single carrier per transponder' or in 'multiple-carriers per transponders' (FDM). On a transponder with the single carrier configuration, the transmission rate R_s can be adapted to available bandwidth (at -3 dB) in order to obtain the maximum transmission capacity compatible with the acceptable signal degradation due to transponder bandwidth limitations. In the multiple-carrier configuration (Frequency Division Multiplexing), the symbol rate R_s must be adapted to the BS (Broadcasting Services) frequencies interval so as to optimize transmissive capacity while keeping the mutual interferences between adjacent carriers at acceptable level.

Figure 3.14 shows the performance achieved by the modem architecture with respect to the unconstrained (from modulation levels and block length of codes) Shannon limit. The ideal $\frac{E_b}{N_0}$ ratios in Table 3.7 for each operating mode has been

[7]We remind that the performance over AWGN channels represents in communications an upper bound to the performance over more realistic channels. For this reason, comparison of the performance between two systems can be accomplished without any loss of generality over an AWGN environment. Comparisons under gaussian hypotheses are expected to be virtually the same over any non-gaussian environment/channel.

derived from the ideal $\frac{E_s}{N_0}$ ratios by using the following dB-relation

$$\frac{E_b}{N_0} = \frac{E_s}{N_0} - 10\log_{10}(\eta_{CM}) \tag{3.20}$$

where η_{CM} is the ideal joint efficiency of modulation and coding adopted to transmit symbols.

Figure 3.15 shows, over the plane C/N – spectral efficiency, the overall performance compared to either the constrained Shannon bounds or the DVB-S performance. The gain of DVB-S2 with respect to DVB-S in terms of C/N, for a given spectral efficiency, remains virtually constant, around $2 - 2{,}5\,\text{dB}$ [7, 6].

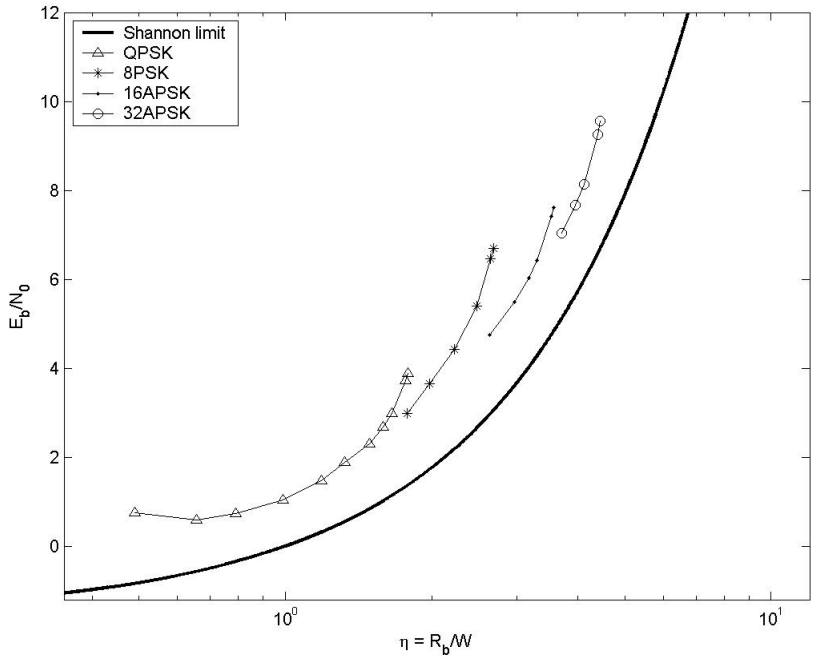


Figure 3.14: Performance of DVB-S2 for each modulation format and coding rate at PER $= 10^{-7}$ with respect to absolute Shannon limit

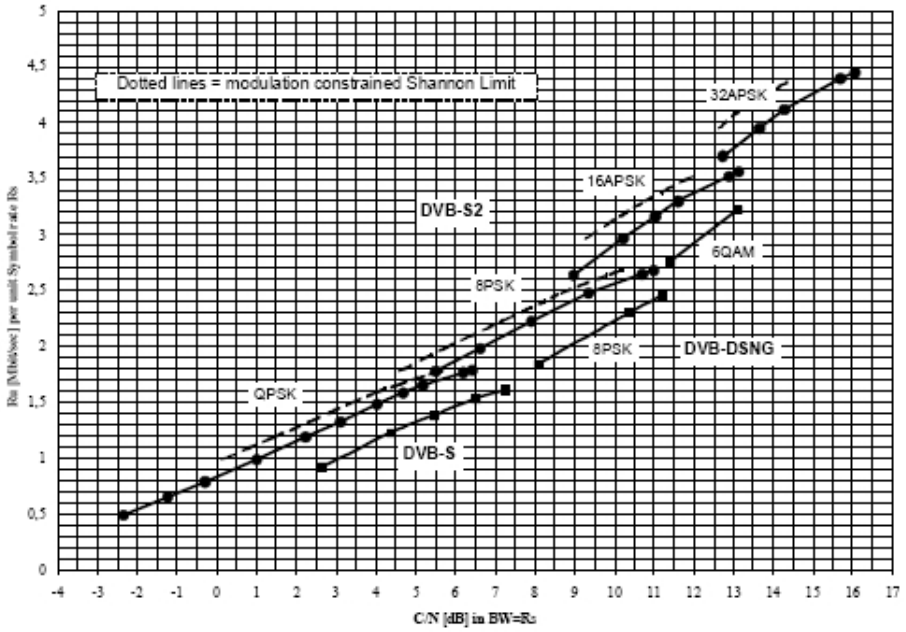


Figure 3.15: Performance Shannon limits constrained

Modulation	Coding rate (LDPC)	Spectral efficiency	Ideal E_s/N_0 for normal FECFRAME	Ideal E_b/N_0 for normal FECFRAME
QPSK	1/4	0,490243	-2,35	0,75
QPSK	1/3	0,656448	-1,24	0,59
QPSK	2/5	0,789412	-0,3	0,73
QPSK	1/2	0,988858	1	1,05
QPSK	3/5	1,188304	2,23	1,48
QPSK	2/3	1,322253	3,1	1,89
QPSK	3/4	1,487473	4,03	2,31
QPSK	4/5	1,587196	4,68	2,67
QPSK	5/6	1,654663	5,18	2,99
QPSK	8/9	1,766451	6,2	3,73
QPSK	9/10	1,788612	6,42	3,89
8PSK	3/5	1,779991	5,5	3,00
8PSK	2/3	1,980636	6,62	3,65
8PSK	3/4	2,228124	7,91	4,43
8PSK	5/6	2,478562	9,35	5,41
8PSK	8/9	2,646012	10,69	6,46
8PSK	9/10	2,679207	10,98	6,70
16APSK	2/3	2,637201	8,97	4,76
16APSK	3/4	2,966728	10,21	5,49
16APSK	4/5	3,165623	11,03	6,03
16APSK	5/6	3,300184	11,61	6,42
16APSK	8/9	3,523143	12,89	7,42
16APSK	9/10	3,567342	13,13	7,61
32APSK	3/4	3,703295	12,73	7,04
32APSK	4/5	3,951571	13,64	7,67
32APSK	5/6	4,11954	14,28	8,13
32APSK	8/9	4,397854	15,69	9,26
32APSK	9/10	4,453027	16,05	9,56

Table 3.7: DVB-S2 performance at PER = 10^{-7}

Chapter 4

BCH Encoding Algorithms for DVB-S2 Digital Transmissions

4.1 Encoding Algorithm Description

As introduced in Section , the systematic encoding of a message

$$\mathbf{m} = (m_{k_{\mathrm{BCH}}-1}, m_{k_{\mathrm{BCH}}-2}, \ldots, m_1, m_0) \tag{4.1}$$

here expressed in vectorial form, provides a codeword

$$\mathbf{c} = (m_{k_{\mathrm{BCH}}-1}, m_{k_{\mathrm{BCH}}-2}, \ldots, m_1, m_0, d_{n_{\mathrm{BCH}}-k_{\mathrm{BCH}}-1}, \ldots, d_0) \tag{4.2}$$

composed of $r = n_{\mathrm{BCH}} - k_{\mathrm{BCH}}$ redundancy bits, aimed at protecting information messages, in the least significant positions. Codewords in systematic form have in the most significant positions the exact replica of information bits.

According to the cyclic code theory (see Appendix B), the systematic encoding of messages can be carried out through these three steps:

1. Multiplication of the message polynomial $m(x)$ by x^r. It can be easily realized through a left shift of the information bits by r positions and a zero padding of the least significant r bits.

2. Division of this vector by the polynomial generator. Coefficients of the remainder, expressed in polynomial form

$$d(x) = d_{r-1}x^{r-1} + d_{r-2}x^{r-2} + \ldots + d_1x^1 + d_0 \tag{4.3}$$

represent the r parity bits.

3. Finally, as also stated in Section A.1,

$$x^r m(x) - d(x) \tag{4.4}$$

provides a codeword. Furthermore, since we are in $GF(2)$ and thus $-d(x) = d(x)$, the expression (4.4) can be rewritten as

$$x^r m(x) + d(x) \tag{4.5}$$

A signal flow diagram of a possible polynomial division implementer is given in Figure 4.1. This kind of architecture is usually called Linear Feedback Shift Register because of the feedback wire connecting the last stage with the first stage of the register. We shall show that the LSFR can succeed in computing the remainder (or parity) bits. This linear system provides instant-by-instant quotient and remainder bits together.

It is possible to demonstrate that this LFRS yields at each sampling instant the temporary remainder bits, which in turns have to be updated at the next time instant. Let us give an intuitive proof of this only focusing on the remainder bits (state) evolution of LFSR. Hereafter we will work only with polynomial coefficients in GF(2), even if generalizations in other base fields[1] are straightforward (e.g. see [9, 21, 15, 12]) .

The binary digits in register, after a certain number of computation cycles required (by now it does not matter how many cycles are required), represent the remainder of division. It is useful to observe that taking the remainder after division by $g(z) = z^4 + z^2 + 1$ is as imposing $z^4 = z^2 + 1$[2]. This operation is performed by the feedback and taps (i.e. those represented as g_1, g_{r-1} in Figure 4.1) of the LFSR.

Since degree of $g(z)$ is 4, the shift register length must be 4 (the remainder of division is a polynomial of degree $r = 3$, which thus have 4 coefficients) as well as the second taps (i.e. g_2) must be enabled. If the message, from MSB to LSB, $m(z) = z^8 + z^4$ enters the first sum (modulo 2) node, we have a delay line between entering bit and bit in x_3 equals to 4. This means that by the feedback wire and taps enabling/disabling all the polynomial coefficients are relatively summed with those of the same degree. When, for example, a z^8 generates the partial result $r(z) = (1 + z^2)$ corresponding to $z^4(1 + z^2)$ at the fifth time instant, the entering coefficient is relevant to degree 4 and then must be summed with the one carried by the feedback wire in order to update x_0.

[1] Very synthetically, multiplier and adder in LFSR shown in Figure 4.2 have to implement these operation in Galois arithmetic.

[2] Over the real field, for example, taking the remainder after division by 4 of any number is as imposing $4 = 0$ so that $13 \bmod 4 = 1$ because 13 can be thought as $4 \times 3 + 1$ and, given that $4 = 0$, then the remainder is 1.

Using the equivalent polynomial notation, let us describe the temporal sequence of all the operations performed.

1. All the register flip flops are set to 0. Coefficient of z^8 enter x_0, so determining $\mathbf{x} = (1, 0, 1, 0)$. We can represent the polynomial division together with digits in the flip flops by the following expression
$$(r_1(z) = 1 + 0 \cdot z^1 + 0 \cdot z^2 + 0 \cdot z^3) \cdot z^8$$
where $r_i(z)$ represent the partial remainder at i-th iteration.

2. $$(r_2(z) = 0 + 1 \cdot z^1 + 0 \cdot z^2 + 0 \cdot z^3) \cdot z^7 + (m_2(z) = 0 \cdot z^7)$$

3. $$(r_3(z) = 0 + 0 \cdot z^1 + 1 \cdot z^2 + 0 \cdot z^3) \cdot z^6 + (m_3(z) = 0 \cdot z^6)$$

4. $$(r_4(z) = 0 + 0 \cdot z^1 + 0 \cdot z^2 + 1 \cdot z^3) \cdot z^5 + (m_4(z) = 0 \cdot z^5)$$

5. Here we have $(r_5(z) = 1 + 0 \cdot z^1 + 1 \cdot z^2 + 0 \cdot z^3) \cdot z^4 + (m_5(z) = 1 \cdot z^4)$ and then
$$(r_5(z) = 0 + 0 \cdot z^1 + 1 \cdot z^2 + 0 \cdot z^3) \cdot z^4$$

6. $$(r_6(z) = 0 + 0 \cdot z^1 + 0 \cdot z^2 + 1 \cdot z^3) \cdot z^3 + (m_6(z) = 0 \cdot z^3)$$

7. $$(r_7(z) = 1 + 0 \cdot z^1 + 1 \cdot z^2 + 0 \cdot z^3) \cdot z^2 + (m_7(z) = 0 \cdot z^2)$$

8. $$(r_8(z) = 0 + 1 \cdot z^1 + 0 \cdot z^2 + 1 \cdot z^3) \cdot z^1 + (m_8(z) = 0 \cdot z^1)$$

9. $$(r_9(z) = 1 + 0 \cdot z^1 + 0 \cdot z^2 + 0 \cdot z^3) \cdot z^0 + (m_9(z) = 0 \cdot z^0)$$
$r(z) = r_9(z)$ corresponding to $\mathbf{x} = (1, 0, 0, 0)$ is also the result of division by $g(z)$.

The key observation to be made in order to understand the encoding process is interpreting each delay tier as the degree of a polynomial (like z-transform).

For the architecture in Figure 4.2, analogous observations upon delay tier between input and output can be made. Here the entering coefficients are summed with the maximum degree coefficients of the partial remainder $r_i(z)$ at each clock cycle, and thus the delay tier is exactly 0. If the result of this sum is 1 then the feedback is enabled, else it does not. As we shall show in the next section, the choice of feeding the shift register by the right-hand side allows us to achieve better performance.

4.2 Serial Architectures

A problem not yet dealt with is how (and when) the parity bits, once computed, can be extracted from the register. As shown in the previous section, LFSR in Figure 4.1 gives the parity bits as all the bits of message-zero-padded, $m(x)x^r$, enter the fist sum (modulo 2) node. Therefore, provided that the remainder after division can be fetched and stored in a single clock tick, this LFSR takes n_{BCH} clock cycles to yield the wanted result.

It is interesting to notice that only at a certain discrete time instant (they furthermore fall periodically) the registers of LFSR contain redundancy bits representing the remainder after division. This drawback prevents this type of architecture from working serially because results of division algorithm have to be properly fetched and stored in any other register when the parity bits are ready.

As just noticed, architecture in Figure 4.1 is slightly unsuitable for a serial systematic encoder since results of divisions (redundancy bits) have to be read in parallel at every $(j+1)n_{\mathrm{BCH}}$ instant for $j = 0, 1, 2 \ldots$. This can be technically performed, but however the architecture would not be serial anymore. Nevertheless, these facts will become quite interesting in parallelization of the architecture.

On the other hand, a mere serial hardware architecture is illustrated in Figure 4.2. Differently from above, here the encoder is fed by the opposite side, thus immediately enabling the feedback, allowing the structure so devised to yield the remainder at every $(j+1)k_{\mathrm{BCH}}$ instead of $(j+1)n_{\mathrm{BCH}}$ for $j = 0, 1, 2 \ldots$. This permits to read serially parity bits and, besides, set (each register is reset) the device so as to make it ready to encode properly next incoming bit streams. Serial encoding in systematic form is accomplished (see Figure 4.2) by:

Systematic Bits Transmission and Parity Computation For clock cycles 1 to k_{BCH}, the information bits are transmitted in the natural order (switch S2 in position 2), while also feeding the shift register, to allow parity bits to be calculated in the Linear Feedback Shift Register (LFSR) (to that purpose, switch S1 must be on, activating the feedback mode).

Parity Bits Transmission For clock cycles $k_{\mathrm{BCH}} + 1$ to n_{BCH}, the parity bits in the LFSR are transmitted (switch S2 in position 1) and the LFSR feedback mode is switched off (by setting S1 off).

From the above timing consideration, it follows that the latter architecture, spending k_{BCH} clock cycles against n_{BCH}, performs more efficiently encoding in case of parallel fetching of the parity bits, as we shall see later on.

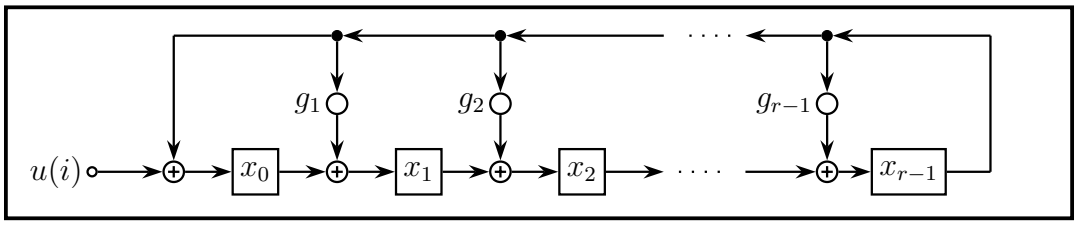


Figure 4.1: Linear Feedback Shift Register architecture implementing the polynomial division in n_{BCH} clock cycles

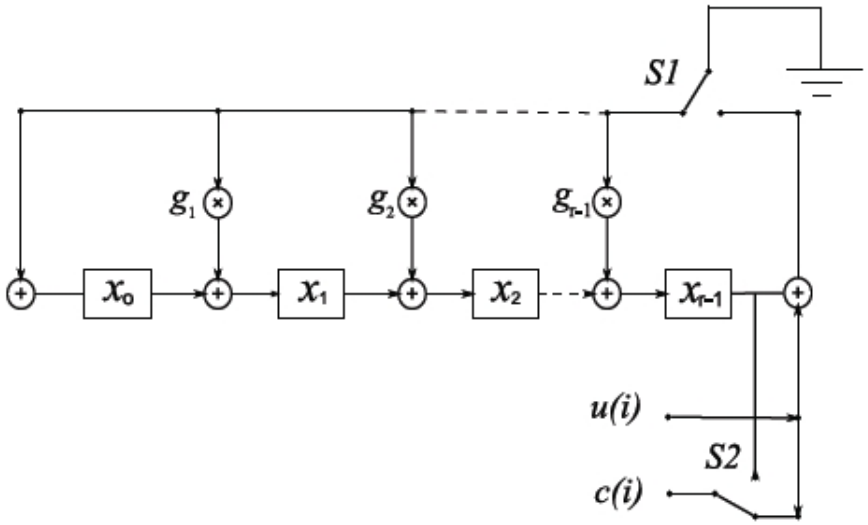


Figure 4.2: Serial encoder architecture based on LFSR (systematic encoding): it implements the systematic encoding of any codeword in n_{bch} clock ticks. At the end of each message encoding, encoder does not require a reset of the state register and thus is ready to immediately encode a new incoming message.

4.3 Encoding Algorithms for Parallel Architectures

4.3.1 Modelling System

From the theory of control we know that any linear system, fed by an input signal $\mathbf{u}(i)$ can be described by the following system of linear equation

$$\mathbf{y}(i) = \mathbf{C}\mathbf{x}(i) + \mathbf{D}\mathbf{u}(i) \tag{4.6a}$$

$$\mathbf{x}(i+1) = \mathbf{A}\mathbf{x}(i) + \mathbf{B}\mathbf{u}(i) \tag{4.6b}$$

where (4.6a) expresses its outcomes $\mathbf{y}(i)$ at each sampling (discrete) instant, while (4.6b) describes its trajectory of state $\mathbf{x}(i)$, i.e., the state evolution of the linear system in question. Without descending in further details, we shall particularize both equations for the linear feedback shift register shown in Figure 4.1 and, starting from them, find equations relevant to an architecture with a p degree of parallelism. We shall notice that they will perfectly match with system (4.6).

The Linear Feedback Shift Register (LFSR) depicted in Figure 4.1 is a linear system, and thus can be described by the well known output equation expressing $y(i)$, the scalar output, and the state transition equation expressing the vector state $\mathbf{x}(i)$ as a function of the discrete time i

$$y(i) = \mathbf{c}\mathbf{x}(i) + d \cdot u(i) \tag{4.7a}$$

$$\mathbf{x}(i+1) = \mathbf{A}\mathbf{x}(i) + \mathbf{b}u(i) \tag{4.7b}$$

In particular, for the desired parallel BCH encoding procedure, we are only interested in the (state) vector

$$\mathbf{x}(i) = \begin{pmatrix} x_0(i) & x_1(i) & \dots & \dots & x_{r-1}(i) \end{pmatrix}^{\mathrm{T}} \tag{4.8}$$

and in its temporal evolution/transition because $\mathbf{x}(i)$ represents the parity bits after the required number of computation cycles. Thus, hereafter we shall focus only on (4.7b). The scalar $u(i)$ indicates the input at the sampling instant i, as put in evidence in Figure 4.1 and Figure 4.2. The matrix $\mathbf{A}$ models the state evolution of the system in absence of external stimulus (i.e., $u(i) = 0$), and thus is usually called the state transition matrix. For analogous reasons, the column vector $\mathbf{b}$ is called as the input-to-state transfer relationship.

The form of $\mathbf{A}, \mathbf{b}$ depends uniquely on the position of taps and connections in the LFSR. In other words, that form is strongly dependent on the selected architecture implementing the encoding algorithm. More precisely, the matrix $\mathbf{A}$ depends on the taps collocations of LSFR, whereas vector $\mathbf{b}$ depends on where the input/s is/are connected to the system.

Now, referring to the LFSR architecture depicted in Figure 4.1, which can yield the redundancy bits after n_{BCH} clock cycles, we can write, on the one hand,

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & \dots & 0 & g_0 \\ 1 & 0 & \dots & 0 & g_1 \\ \vdots & 1 & \dots & 0 & g_2 \\ & \vdots & \dots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & g_{r-1} \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{pmatrix} \tag{4.9}$$

where $\mathbf{A}$, r by r state transition square matrix, contains in the last column the polynomial generator coefficient except for its coefficient of maximum degree. On

the other hand, column vector $\mathbf{b}$ of length r gives a mathematical representation of the physical input-to-system connections.

As introduced in Section 4.2, the adoption of the architecture in Figure 4.2 permits to save some clock ticks. Not surprisingly, only $\mathbf{b}$ has changed its form whereas the representation of matrix $\mathbf{A}$ given in (4.9) still holds. Since inputs enter now from the right-hand side, we have the new column vector

$$\mathbf{b} = \begin{pmatrix} g_0 \\ g_1 \\ \vdots \\ \vdots \\ g_{r-1} \end{pmatrix} \tag{4.10}$$

which models – as before – the input incidence on the system state.

4.3.2 Parallel Linear System Description

In the previous section we have briefly described a possible model of system which can be easily exploited in parallelizing a general serial architecture with a degree of parallelism p. Starting from (4.7b) and applying instant-by-instant and recursively the following substitutions

$$\begin{aligned} \mathbf{x}(i) &= \mathbf{A}\mathbf{x}(i-1) + \mathbf{b}u(i-1) \\ \mathbf{x}(i-1) &= \mathbf{A}\mathbf{x}(i-2) + \mathbf{b}u(i-2) \\ &\vdots \\ \mathbf{x}(i-p+1) &= \mathbf{A}\mathbf{x}(i-p) + \mathbf{b}u(i-p) \\ \mathbf{x}(i) &= \mathbf{A}\left[\mathbf{A}\mathbf{x}(i-2) + \mathbf{b}u(i-2)\right] + \mathbf{b}u(i) \\ &= \mathbf{A}^2\mathbf{x}(i-2) + \mathbf{A}\mathbf{b}u(i-2) + \mathbf{b}u(k-1) \end{aligned}$$

we can observe that the evolution of this system at a generic sampling instant i depends on

- the state vector relevant to a generic former instant;
- a set of former inputs whose cardinality is equal to the difference among the current time index and the former instant selected.

This is equivalent, from an architectural implementation point of view, in feeding the encoding processor by a set of parallel inputs, associating to each of the multiple input wires a different (multiple) time instant. Hence, parallelization is straightforwardly achieved.

Returning to the above expression, the system evolution for a generic degree of parallelism p is modelled by the following state equation

$$\mathbf{x}(i) = \mathbf{A}^p(i-p) + \sum_{k=0}^{p-1} \mathbf{A}^k \mathbf{b} u(i-k-1) \tag{4.11}$$

Now, setting the column vectors $\mathbf{A}^k\mathbf{b}$ as follows

$$\begin{pmatrix} \mathbf{b} & \mathbf{Ab} & \dots & \mathbf{A}^{p-2}\mathbf{b} & \mathbf{A}^{p-1}\mathbf{b} \end{pmatrix} \tag{4.12}$$

we can define a new r by p $\mathbf{B}_p$ matrix which represent the incidence of the last p inputs

$$\mathbf{u}(ip) = \begin{pmatrix} u(ip-1) & u(ip-2) & \dots & \dots & u\left[p(i-1))\right] \end{pmatrix}^{\mathrm{T}} \tag{4.13}$$

on the system evolution. Considering both (4.12) and (4.13), we can therefore rewrite the state equation (4.11) as

$$\mathbf{x}(ip) = \mathbf{A}^p \mathbf{x}\left[(i-1)p\right] + \mathbf{B}_p \mathbf{u}(ip) \tag{4.14}$$

where the sum $\sum_{k=0}^{p-1} \mathbf{A}^k \mathbf{b} u(i-k-1)$ has just been replaced by the above matrices product.

4.3.3 Matrices Structure and Properties

In every practical case, the wanted degree of parallelism is strictly less than the number of registers containing the remainder bits at the end of every bit stream processing. For this reason, in this section we shall describe the main characteristics of matrices $\mathbf{A}^p$ and $\mathbf{B}_p$ only within the range $0 < p \leq r$ of our interest.

Therefore for a generic degree p (within the above specified range) we have

$$\mathbf{A}^p = \begin{pmatrix} \mathbf{0} & \mathbf{C}_1 \\ \mathbf{I} & \mathbf{C}_2 \end{pmatrix} \tag{4.15}$$

where

- $\mathbf{0}$ represents a null $p \times p$ matrix;
- $\mathbf{I}$ represent a square $r-p$ by $r-p$ identity matrix;
- $\mathbf{C}_1$ is a $p \times p$ matrix where each row represents the combinatorial connections (1 when it is active, 0 otherwise) between last and first p bits of vector $\mathbf{x}(i)$;
- $\mathbf{C}_2$ is a $(r-p) \times p$ matrix where each row represents the combinatorial connections between last p bits and the remaining bits of the vector $\mathbf{x}(i)$

Differently from $\mathbf{A}^p$ matrix, $\mathbf{B}_p$ as well as $\mathbf{b}$ (as just said in Section 4.3.1) is strictly dependent on where inputs are. Referencing to the LFSR architecture depicted in Figure 4.1, we have therefore

$$\mathbf{B}_p = \begin{pmatrix} \mathbf{I} \\ \mathbf{0} \end{pmatrix} \tag{4.16}$$

in case on which the feedback starts at least (depends on the position of the first one in the message) after $\frac{r}{p}$ clock ticks (supposing that p is a divisor of r). Identity matrix has size $p \times p$. For example, for a BCH$(7, 15)$ with a degree of parallelism $p = 5$ [3] we get

$$\mathbf{A}^5 = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix} \qquad \mathbf{B}_5 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{4.17}$$

As just shown, the structure of above matrix $\mathbf{B}_5$ is quite trivial. This result is a direct consequence of the way by which we have defined $\mathbf{B}_p$ in (4.12). Because of simple nature of vector $\mathbf{b}$ (which has only a one on the first row), the products between it and increasing powers of $\mathbf{A}$ can be obtained by selecting the first column of the first p powers $\mathbf{A}^0 = \mathbf{I}, \mathbf{A}^1, \mathbf{A}^2, \ldots \mathbf{A}^{p-1}$ of state matrix. In other words, $\mathbf{B}_p$ can be conveniently expressed as

$$\begin{bmatrix} \mathbf{I}(1:,1) & \mathbf{A}(1:,1) & \ldots & \mathbf{A}^{p-2}(1:,1) & \mathbf{A}^{p-1}(1:,1) \end{bmatrix} \tag{4.18}$$

where a notation as $\mathbf{X}(1:,j)$ indicates the j-th column of any matrix. In our case we have then to select each first column of matrices in (4.18).

The matrix $\mathbf{B}_p$ in (4.17) becomes non-structured as the shift register are fed on the opposite side relatively to first and less efficient architecture shown previously (Figure 4.1). In fact, taking into account the new definition (4.10) of column vector $\mathbf{b}$ relevant to the positions of inputs into the system, we get a new $\mathbf{B}_p$ matrix which, differently from the previous, does not show any property of regularity. In other words, this new matrix has no longer the trivial form shown in (4.17).

[3] The degree of parallelism must be a divisor of numbers of clock ticks to encode a codeword. This architecture spends a number of ticks equal to 15 (codeword length) and therefore p can be 5.

4.4 Some Considerations

From an hardware point of view, following considerations can be made:

1. since codeword length compliant with DVB-S2 standard is long (either short or long FECFRAME), a serial encoder (e.g., that depicted in Figure 4.2) may be rather inefficient and slow;

2. parallel architectures are more flexible solutions when different t-error correction BCH codes are needed. Since the standard provides different t (for BCH) that can be switched on a frame basis, encoder is required to be versatile;

3. the overall architecture (inner LDPC encoder in parallel implementation) has to be taken into account and a parallel encoder make the interface design simpler.

On the contrary, serial architectures, being based on an elementary LSFR substructure, are rather simpler than their parallel version. But, for the reasons enumerated above, they have not been taken in consideration to design the encoding section of DVB-S2.

Chapter 5

Hardware Implementation of BCH Encoder

5.1 FEC Encoding Section

The parallelism of the input memory of the LDPC encoding section suggests a $p = 8$ degree of parallelism for BCH architecture. First, because LDPC input memory requires 360 bits and LDPC encoder processes the same number of bits at every clock ticks (i.e., it has 360 as degree of parallelism). Hence, p should be a divisor of 360. Second, p should be even a divisor of each BCH block length (n_{bch}) associated to LDPC coding rates, provided by DVB-S2. Furthermore, BCH serial architectures based on LFSRs are, of course, simple to implement, but generally they are very slow (i.e., they reach typically a lower throughput). Therefore, in order to best match BCH encoding speed with the frequency requirements imposed by the overall designed TX DVB-S2 section, we have chosen a degree of parallelism equal to 8.

Encoded bits, passing through an interface circuit, enter LDPC input memory on a byte basis. The interface allows each codeword coming from BCH encoder to be read in the proper order and format (recall that a systematic code is required as output). A block diagram of the overall FEC section is depicted in Figure 5.1.

As we shall detail in the next section, the interface between BCH and LDPC is constituted by a simple 8 bits multiplexer together with a suited circuitry to store and download the parity bits after each computation cycle. This interface addresses properly encoded bits into LDPC input memory and it must be driven by a control logic based on clock cycles required (see Table 5.1) to provide a codeword.

Another important question to raise up is certainly giving the architecture a flexible structure so as to succeed in dealing with each coding rate provided by DVB-S2 FEC. A more versatility, as we shall see, requires an additional usage of memory: as a matter of fact, some additional coefficient of matrices mentioned in

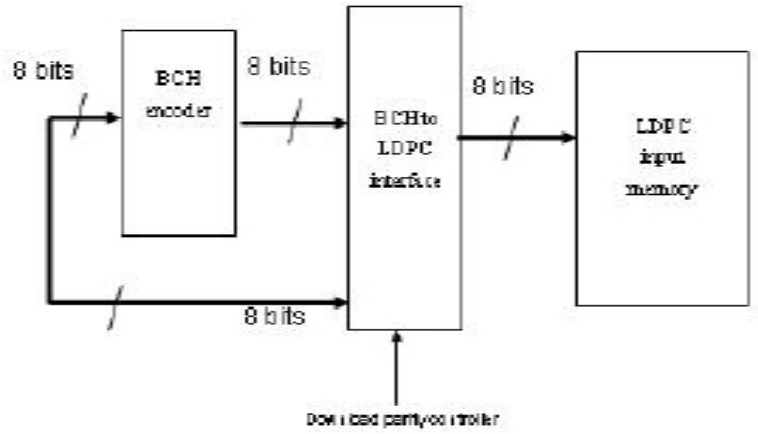


Figure 5.1: Block diagram of the overall FEC tx section

the previous chapter must be stored into some LUTs (Look Up Tables).

BCH Uncoded Block	BCH Coded Block N_{BCH}	Ticks to transmit informative bits	Parity bits download	Total clock cycles
16008	16200	2001	24	2025
21408	21600	2676	24	2700
25728	25920	3216	24	3240
32208	32400	4026	24	4050
38688	38880	4836	24	4860
43040	43200	5380	20	5400
48408	48600	6051	24	6075
51648	51840	6456	24	6480
53840	54000	6730	20	6750
57472	57600	7184	16	7200
58192	58320	7274	16	7290

Table 5.1: Clock cycles required to provide each codeword for each operating mode of DVB-S2

5.2 Encoder Description

A block diagram of BCH encoder architecture is shown in Figure 5.4. A brief description on its functioning is given below.

- All the informative k_{bch} bits enter in parallel (p bits, i.e., 8 at once), from MSB to LSB, all 192 combinatorial blocks, which perform logic function indicated by $\mathbf{B}_p$ matrix. In other words, each block carries out a row by columns product (over $GF(2)$) between the i-th row of $\mathbf{B}_p$ the p inputs. Since operations are in GF(2), sums are implementable by XOR gates, multiplications by AND gates. Figure 5.3 gives a schematic illustration of each combinatorial networks processing p inputs.

- Each flip-flop labelled by x_i represents a single bit of state vector $\mathbf{x}$ of the system and after exactly $\frac{k_{\text{bch}}}{p}$ clock ticks it contains the result of division algorithm.

- Combinatorial networks toward the BCH-to-LDPC interface (i.e, on the output side) perform the feedback: each XOR gate is fed by the last p bits of register state $\mathbf{x}$ (from x_{184} to x_{191}) passing through output-combinatorial network described by the two sub-matrices $\mathbf{C}_1$ and $\mathbf{C}_2$ (see Section 4.3.3). More in details, expression (4.15) in Section 4.3.3, showing an high regularity of $\mathbf{A}^p$, allows to split computation of the first term in (4.14) (i.e, $\mathbf{A}^p\mathbf{x}\,[(i-1)p]$) in two contributes

  1. row by column products involving last p bits of $\mathbf{x}$ and therefore last p columns of matrix $\mathbf{A}^p$. In other words, the combinatorial networks near to BCH-LDPC interface perform the following product

$$\begin{pmatrix} \mathbf{C}_1 \\ \mathbf{C}_2 \end{pmatrix} \begin{pmatrix} x_{184}(i) \\ \vdots \\ x_{191}(i) \end{pmatrix} \tag{5.1}$$

  2. row by column product involving (starting from bit x_8 of vector $\mathbf{x}$ which correspond to row 9 of matrix $\mathbf{A}^p$) first $n-k-p$ bits of vector $\mathbf{x}$ is realized by sum modulo 2 (i.e., XOR gates) nodes on the right of the combinatorial networks. Figure 5.2 gives a schematic representation of this combinatorial networks together with modulo 2 sum nodes.

- Logic functions relevant to sub-matrix $\mathbf{I}$ of $\mathbf{A}^p$ are implemented by XOR gates at the output of combinatorial networks connected to the last p bits of the state vector.

- At the end of each computation cycle, the encoder register must be reset.

Due to intrinsic simplicity of matrices $\mathbf{A}^p$ and $\mathbf{B}_p$ (they are only composed by zeroes and ones), the two combinatorial networks can be implemented in a very simple way by programmable XOR gates with eight inputs. More in detail, each coefficient of the above two matrices says which wires have to be connected to each combinatorial network (XOR).

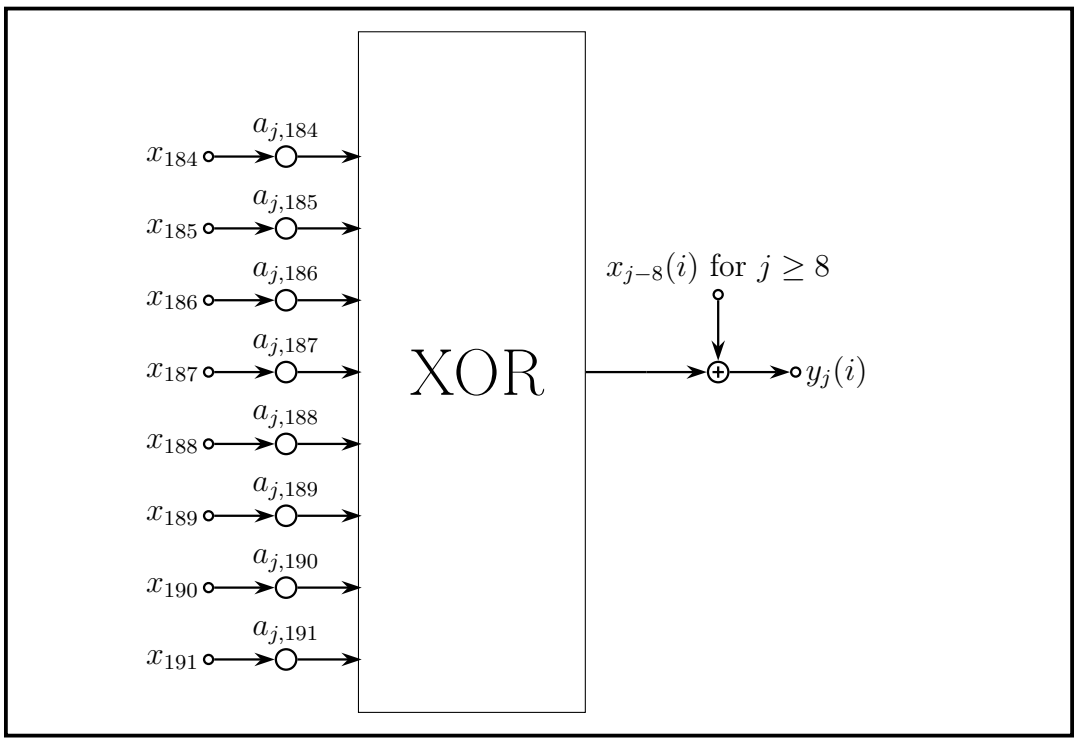


Figure 5.2: Block scheme of the combinatorial networks acting on the last p bits of $\mathbf{x}$. $a_{j,l}$ is the coefficient at j-th row and l-th column of $\mathbf{A}^8$ matrix

5.3 Dealing with Each Error Protection

Architecture we have shown as far can only address the greater t error protection modalities. However, DVB-S2 has been designed to operate in ACM (Adaptive Coding and Modulation) mode, so that a flexible architecture, capable of changing his behavior on a frame basis, would be more desirable.

From our analysis of the two matrices, we have reach the conclusion that they are very regular and, even changing the polynomial generator of the code, this

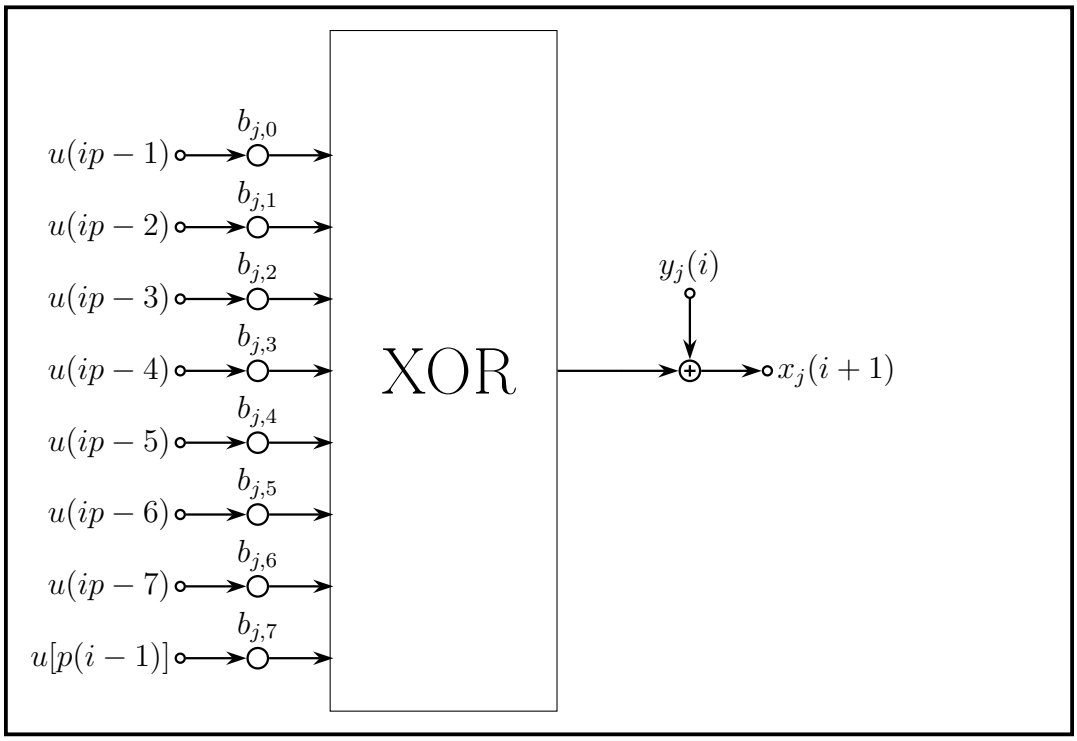


Figure 5.3: Block scheme representing the combinatorial networks acting on the p bit of input. $b_{j,l}$ is the coefficient at j-th row and l-th column of the $\mathbf{B}_8$ matrix

interesting property still holds. The degree of parallelism p, once set by design, cannot vary. Henceforth all our considerations will be constrained to a specific degree of parallelism, $p = 8$, even though the method we are going to show is general at all.

BCH of DVB-S2 provides three protection level $t = 8, 10, 12$ associated to the following different generators

$$g_{t_{12}}(x) = g_1(x)g_2(x)\dots g_{12}(x) \tag{5.2}$$
$$g_{t_{10}}(x) = g_1(x)g_2(x)\dots g_{10}(x) \tag{5.3}$$
$$g_{t_8}(x) = g_1(x)g_2(x)\dots g_8(x) \tag{5.4}$$

where the above polynomials are in Table 3.4. According to the decrease of the degree of polynomial generator, also the number r of redundancy bits decreases and thus the number of FFs illustrated in Figure 5.4 would be oversized. This implies that, $\mathbf{x}$ being changed, size of matrices $\mathbf{A}^8$ and $\mathbf{B}_8$ has to change. In Section 4.3.3 we have shown that, for a reasonable level of parallelism, matrix $\mathbf{A}^p$ shows some regularity. In practice, its structure shown in (4.15) does not change from t_8 to t_{12},

although its coefficients are, of course, subject to variations. Same consideration can be made upon $\mathbf{B}_p$ matrix, which however shows no regularity property for this kind of architecture.

Figure 5.3 and Figure 5.2 clearly show that these kind of combinatorial network are programmable by means of their coefficients, determining which wire has to be connected to the 8-inputs XOR gate and which not. Imposing, for example, that all of these coefficients are forced to zero, we would get the result of having inhibited the j-th combinatorial network. This is exactly what we can do to deal with all the protection level t.

In particular, either $\mathbf{A}^8$ or $\mathbf{B}_8$ can be embedded into the their largest matrices of size, respectively, 192×192 and 192×8 in this way:

Medium Protection Level ($t = 10$) Matrices $(\mathbf{A}^8)_{\mathrm{o}}$ and $(\mathbf{B}_8)_{\mathrm{o}}$ oversized (they should be 160×160 and 160×8 respectively) turn out to be as follows

$$(\mathbf{A}^8)_{\mathrm{o}} = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (\mathbf{A}^8)_{\mathrm{t}_{10}} \end{pmatrix} \qquad (\mathbf{B}_8)_{\mathrm{o}} = \begin{pmatrix} \mathbf{0} \\ (\mathbf{B}_8)_{\mathrm{t}_{10}} \end{pmatrix} \tag{5.5}$$

This corresponds to inhibit the first 32 couples of combinatorial networks since their coefficients are all nulls. Thus, all the first 32 FFs (from x_0 to x_{31}) of the BCH encoder will contain always zeroes.

Low Protection Level ($t = 8$) The oversized matrices are build in the same manner and now the first 64 couples of combinatorial networks are inhibited since their coefficients are all nulls.

5.4 Interface and Parity Bits Extraction

The interface between BCH and LDPC encoder can be implemented as illustrated in Figure 5.5. The parity bits, once computed (at n_{bch} clock tick), can be saved (in a single clock cycle) in a shift register architecture, which has been called in Figure 5.5 192 (or rather, up to 192) to 8 bit converter. In fact, its task is formatting data in the DVB-S2 format. Furthermore, this kind of architecture allows to write all the parity bits extracted from BCH encoder (in a single clock tick) into LDPC input memory with a degree of parallelism equal to 8[1].

Further detailing, the converter is composed by 8 shift register blocks with size equal to 24 bits. In a preliminary fase all the parity bits computed by the BCH encoder are stored in these blocks following the labelling indicated in Figure 5.6 (to

[1]Note also that $p = 8$ is not only a divisor for 360, but also for 192($t = 12$), 160($t = 10$) and 128($t = 8$)

succeed in storing these bits, MUXs between each flip flop can be used). Afterward, the converter works as a shift register, thus allowing to download data eight at once, i.e., with a degree of parallelism equal to 8.

To summarize, let us see, in order for each iteration, the operations carried out by this specific interface:

1. Once the parity bits have been computed, they are stored in another shift register according to the bit-to-bit mapping illustrated in Figure 5.6. MUXs between a FF and its neighbor allow to store bits all at once (in parallel) as they are switched on wires carrying results of encoding (these wires may be directly connected to the BCH encoder register).

2. As MUXs are switched on wires interconnecting FFs each other, all the bits previously stored, shifting along FFs, are carried by some 'strategic' wires on the 8 bits bus, which conveys parity bits toward the output multiplexer. Practically, the register is working as a shift register.

3. The output multiplexer has two inputs buses connected to: informative bits and redundancy bits. MUX should be driven by a proper control logic on the operating mode basis.

5.5 Frequency and Memory Requirements

Taking into account LDPC outer encoder frequency requirements and imposing a data rate equal to 1 Gbps, in the worst case, that is, when $t = 8$ and LDPC code rate is equal to 9/10, we obtain the maximum frequency requirement so as to guarantee this challenging performance:

$$(f_{\text{clk}})_{\text{MAX}} = \frac{frame}{s} \cdot 7290 = 125\,\text{MHz} \tag{5.6}$$

Concerning to memory requirement either, the architecture proposed takes six (two per each level of BCH protection parameter t) LUTs to store coefficients used by the combinatorial networks. As a matter of fact, storing of the $\mathbf{C}_1$ and $\mathbf{C}_2$ sub-matrices (recall that they represent two sub-blocks, whose aggregated size is 192×8, of the $\mathbf{A}^8$ matrix) coefficients as well matrix $\mathbf{B}_8$, whose size is 192×8, coefficients is required. To provide an encoder capable of dealing with each t-error protection level, those matrices have to be saved in LUTs for each t. The size of each LUT is equal to 192 Bytes and thus the BCH encoder, apart from additional memory required by the BCH-LDPC interface and the encoder state register, requires an amount of memory equal to

$$2 \cdot 3 \cdot 192\,\text{B} = 1{,}125\,\text{KiB}. \tag{5.7}$$

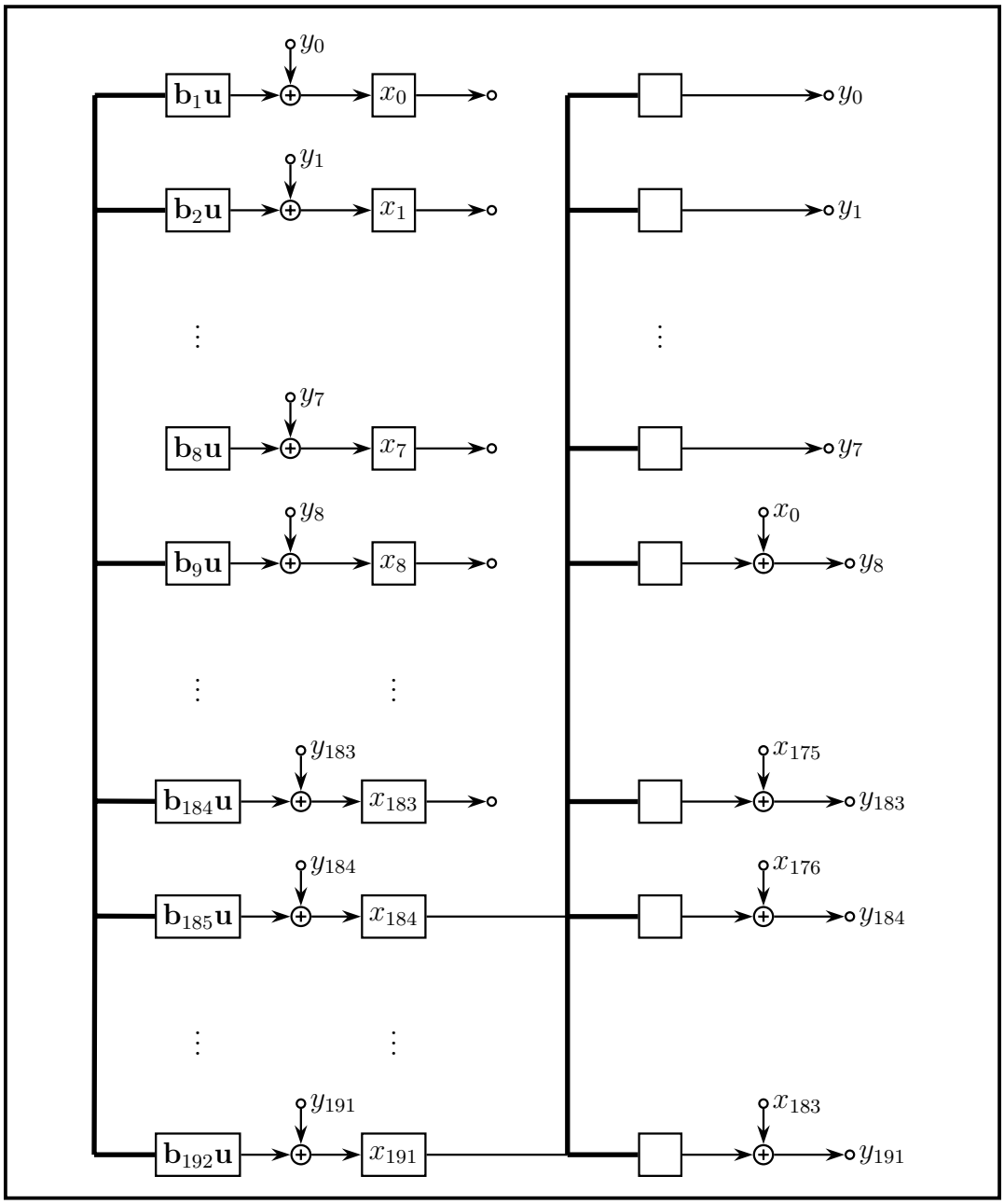


Figure 5.4: Architecture oriented to digital hardware implementation. Each combinatorial network on the left-hand side implements a row by column product of $\mathbf{B}_8\mathbf{u}$. Each row of matrix $\mathbf{B}_8$ is indicated as $\mathbf{b}_j$ with $0 \leq j < 192$. Each combinatorial network on the right-hand side implements a row by column sub-product in (5.1). The 8 bit bus on the input side carries message bits to be encoded.

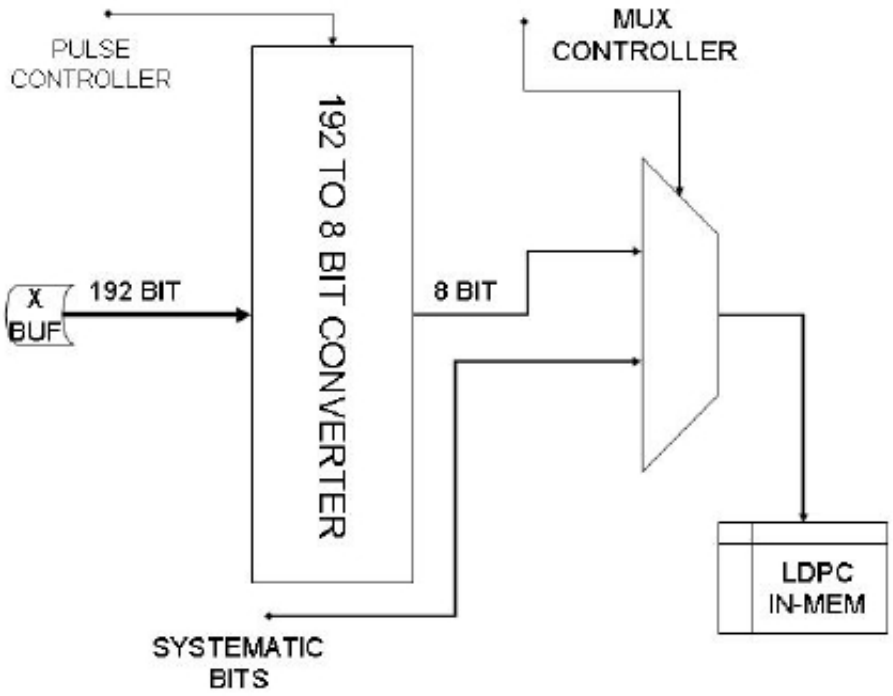


Figure 5.5: Interface architecture. Parity bits are loaded into the 192 to 8 bits converter.

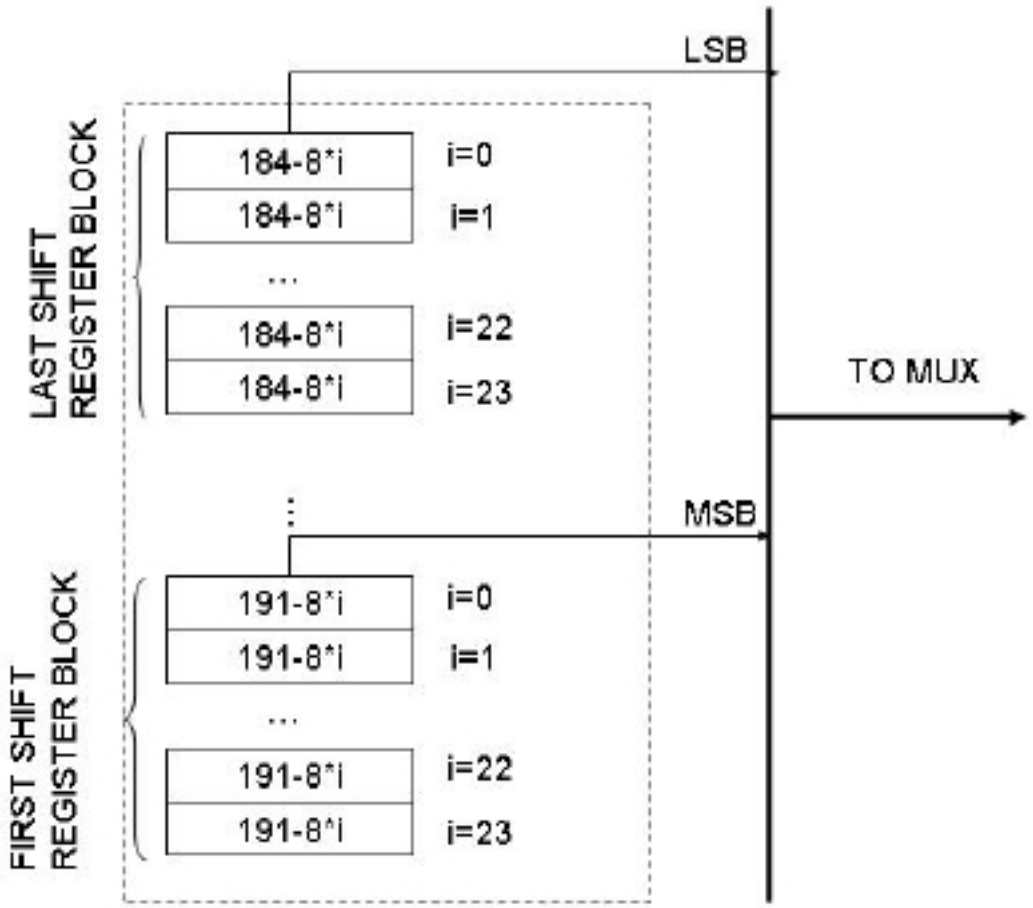


Figure 5.6: 192 to 8 bits converter architecture. Values into each FF represent the index mapping between the BCH encoder register and the interface register, used to extract and format data.

Chapter 6

Software Package for VHDL Validation

6.1 Software Implementation of Serial Encoder

This section describes the software implementations of the architecture illustrated in Figure 4.2. Recall that only this architecture is actually serial since the other one (depicted in Figure 4.1), which computes parity bits in n_{bch} clock ticks, needs a parallel fetching of the encoding result and a reset of the register at the end of each computation cycle[1].

The software implementation shown below simulates the serial architecture depicted in Figure 4.2. For the first k clock ticks, the informative bits exits while the feedback loop is enabled. From k to n instead, the feedback loop is disabled and then all the parity bits ready to be fetched are carried (serially) to the output while zeros are going to be stored into each stage of the shift register, thus resetting the register. This is certainly an advantage with respect to architecture depicted in Figure 4.1, which requires a parallel fetching of the parity bits together with, in turn, a reset of the shift register.

To simulate the two different behaviors of the serial encoder, which for k clock cycles yields the bits of message while for the next one produces the parity bits, a integer `encStep` counter has been employed. It follows a useful description of the used variable:

- `ticks` is a function parameter and refers to the numbers of iteration/clock-ticks which have to be simulated; `m[], out[]`, other ones function parameters, are vector relevant to input and output. The function `Run` reads from `m[]` and write in turn the result of `ticks` computation cycles.

[1] Clearly, in software, this distinction is not so strict

- Vector of integer `state[]` represent the values contained in each register of the encoder.
- `g[]` represent in vectorial form polynomial generator of the BCH code.
- `r` is the length of the shift register.

```
void BCHenc::Run(int ticks,int *m,int *out)
{
    int i,s, j,k;
    if(encStep == n) {encStep = 0;}
    if(encStep < k){
        for(i = 0; i < ticks; i++, encStep++){
        // If condition is true enable feedback
            if(state[r-1]^m[i]){
      for(j = r-1; j >=1; j--)
        state[j] = state[j-1]^g[j];
      state[0] = 1;
        }
      else{
      // Shifting of bits
      for(j = r-1; j >= 1; j--)
        state[j] = state[j-1];
      state[0] = 0;
      }
    out[i] = m[i];
    }
  }
  else if(encStep >=k && encStep < n){
        for(i = 0; i < ticks; i++, encStep++){
            out[i] = state[r-1];
            for(j = r-1; j >= 1; j--)
                state[j] = state[j-1];
            state[0] = 0;
    }
  }
}
```

This piece of software works serially, but the user may define the number of iterations that emulator machine has to perform. Anyway this software emulator, as its more physical version, takes n clock ticks to encode a single codeword.

6.2 Software Implementations of Parallel Encoder

Matrices $\mathbf{A}^8$ and $\mathbf{B}_8$ defined in Section 4.3.3 have been pre-computed via software by a Matlab routine. Useful coefficients of these matrices can be stored in local variables (software) or in LUT (hardware). Note that, concerning the $\mathbf{A}^8$ matrix, the sub-matrices $\mathbf{C}_1$ and $\mathbf{C}_2$ should be stored in a dedicated memory for each t-error correction level to make the architecture flexible.

The first software implementation refers to the slower parallel architecture which spends n clock ticks to provide codewords associated to messages. Pre-computed parts of the matrix $\mathbf{A}^8$ relevant to each operating modes are loaded and saved in memory on (n, k) (only those provided by the standard) basis. Matrix $\mathbf{B}_p$, already defined in (4.16) (see also the example in Section 4.3.3), is trivial and its save in memory can be avoided since that form corresponds to connect the eight inputs to the first XOR stage of the architecture. In other words, the 192 combinatorial

networks on the input side (depicted in Figure 5.4) in this kind of architecture are not necessary. The following function is dedicated to simulate the functioning of combinatorial networks on the output side:

- Feedback combinatorial network acting on the last eight values of the state register. Function `combn(index, n, regold)` implements the product row by column between $\mathbf{C}_1, \mathbf{C}_2$ and $x_{184}, \ldots, x_{191}$.

```
int comb_n(int index,int r, int *reg_old)
{
  int out,f;

  out=0;

  for (f=0; f<P; f++)

  {
    out=out^(Ap_n[index][r-f-1] & reg_old[r-f-1]);
  }

  return(out);
}
```

The first part of function `BCHnclkpar(int n,int k)` is relevant to the error protection level and consequently due to the state vector allocation (recall that register length can be 128, 160, 192 with respect to $t = 8, 10, 12$ possible values). In case of mismatch with the couples n, k, provided by the standard DVB-S2 for the normal FECFRAME, simulation is aborted. The `for` cycle in the middle part of the program updates cyclically the register of the encoder, using the above function implementing each combinatorial networks. Eventually, output is formatted in the systematic form complyant with the DVB-S2 standard requirements.

```
void BCHnclk_par(int n,int k)
{
  int clock_ticks;
  int *reg, *reg_old;

  int input[P]; // parallel input bits

/*  Mode Selection (t-error-correction) */

  switch(n-k) {
  case 192:
    reg = (int*)calloc(n-k,sizeof(int));
    reg_old = (int*)calloc(n-k,sizeof(int));
    break;
  case 160:
    reg = (int*)calloc(n-k,sizeof(int));
    reg_old = (int*)calloc(n-k,sizeof(int));
    break;
  case 128:
    reg = (int*)calloc(n-k,sizeof(int));
    reg_old = (int*)calloc(n-k,sizeof(int));
    break;
  default:
    fprintf(stdout,"Error:simulation aborted!\n");
    fprintf(stdout,"Please insert a n-k couple
                          provided by DVB-S2 FEC\n");
    exit(0);
  }
  /* Computation of clock ticks required
       to compute the remainder after division */
  clock_ticks = n/P;
```

```
    /* Computing remainder */
  int z=0;

  for (int i=0; i<clock_ticks; i++)
  {
    /* refresh of state  */
    for (int m=0; m<n-k; m++)
      reg_old[m]=reg[m];
    ////////////////////////////////////////
    /////// loading of parallel input //////
    for (int count=P-1; count>=0; count--)
    {
      z++;
      input[count] = message[n-z];
    }
    ////////////////////////////////////////////
    /// Computing of next values of state /////
    if (clock_ticks >0)
    {

      for (m=0; m<n-k; m++)
      {
        if (m<P)

          reg[m] = input[m]^comb_n(m,n-k,reg_old);

        else

          reg[m] = comb_n(m,n-k,reg_old)^reg_old[m-P];

      }
    }
    ////////////////////////////////////////////////
  }

    /* Codeword in systematic form */

  for (i=n-1; i>n-k-1; i--)
    codeword[i] = message[i];

  for (i=n-k-1; i>=0; i--)
    codeword[i] =  reg[i];

}
```

The second implementation is connected to the faster architecture (its correspondent serial version is depicted in Figure 4.2) which spends k clock ticks to compute parity bits, saving, compared to the first, r clock cycles for each computation cycle or, i.e., for each encoding cycle. The function `combn(index, n, regold)` implementing the combinatorial networks on the output side is the same of the slower architecture according to what we said in Chapter 4 (i.e. matrix $\mathbf{A}_8$ cannot change). Therefore here we have two functions:

- Function `combc(index, input)` provides the result of row by column product between matrix $\mathbf{B}_8$ and the inputs.
- Function `combn(index, n, regold)` implements the product row by column between $\mathbf{C}_1, \mathbf{C}_2$ and $x_{184}, \ldots, x_{191}$.

```
int comb_c(int index, int *input)
{
  int out,f,ind;

  out=0;

  ind=P-1;

  for (f=0; f<P; f++)
  {
```

```
    out= out ^ ((C[index][f]) & (input[f]));
    ind--;
  }

  return(out);
}


void BCHkclk_par(int n,int k)
{
  int clock_ticks;
  int *reg, *reg_old;
  int offset,m;

  int input[P]; // parallel input bits

    /* Mode Selection (t-error-correction) */

  switch(n-k) {
  case 192:
    reg = (int*)calloc(MAXR,sizeof(int));
    reg_old = (int*)calloc(MAXR,sizeof(int));
    offset = MAXR-192;
    break;
  case 160:
    reg = (int*)calloc(MAXR,sizeof(int));
    reg_old = (int*)calloc(MAXR,sizeof(int));
    offset = MAXR - 160;
    break;
  case 128:
    reg = (int*)calloc(MAXR,sizeof(int));
    reg_old = (int*)calloc(MAXR,sizeof(int));
    offset = MAXR-128;
    break;
  default:
    fprintf(stdout,"Error:encoding aborted!\n");
    fprintf(stdout,"Please insert a n-k
                    couple provided by DVB-S2 FEC\n");
    exit(0);
  }
  /* Computation of clock ticks required
       to compute the remainder after division */
  clock_ticks = k/P;

    /* Computing remainder */

  int z=0;
  for (int i=0; i<clock_ticks; i++)
  {
    for (m=0; m < MAXR; m++)
      reg_old[m]=reg[m];

    for (int count=P-1; count>=0; count--)
    {
      z++;
      input[count] = message[n-z];
    }


    if (clock_ticks>0)
    {

      for (m=0; m< MAXR; m++)
      {
        if (m<P)

          reg[m] = (comb_c(m,input))^(comb_k(m,reg_old));
        else
          reg[m] = (comb_c(m,input))^(comb_k(m,reg_old))^
                              (reg_old[m-(P)]);
      }
    }
  }
    /* Codeword in systematic form */

  for (i=n-1; i>n-k-1; i--)
    codeword[i] = message[i];

  for (i=n-k-1; i>=0 ; i--)
    codeword[i] =  reg[i+offset];

}
```

6.3 Galois Fields Tables

As said in Section 3.3.1, the knowledge of the big field $GF(2^{16})$ where the roots of polynomial generators are is necessary to make error detection and decoding (e.g. by Berlekamp-Massey Algorithm). Galois field associated to this BCH code can be built by using $g_1(x)$ in Table 3.4, which is also the primitive polynomial of $GF(2^{16})$ for the reasons already expressed in Section 3.3.1.

All the $2^{16} - 1$ elements of this field can be obtained by a LFSR structure (implementing the division algorithm as that depicted in Figure 4.1) where each connection/disconnection (since we are over GF(2)) corresponds to coefficients of $g_1(x)$. This LFSR architecture, however, performs all their operations in absence of external stimulus (i.e, the single input $u(i)$ is forced to zero $u(i) = 0$). The state evolution repeats for each multiple of $2^{16} - 1$: if the shift register is initialized by a unitary seed $(0, \ldots, 0, 1)$, then we will find the same seed after $2^{16} - 1$ clock ticks. This means that the initialization seed is a primitive element of GF(2^{16}). Each primitive element can be found by trial end errors:according to Definition A.3.2, when an initialization seed leads to have a 1 at $2^{16} - 1$ clock tick, then a primitive element has been found, otherwise it does not. In our simulation, for the sake of simplicity, $\alpha = 1$ has been used as primitive element.

```
void gfField(int m, // Base 2 logarithm of cardinality of the Field
       int poly, // primitive polynomial of the Field in decimal form
       int ** powOfAlpha, int ** indexAlpha)
{
  int reg,  // this integer of 32 bits, masked by a sequence of m ones,
        // contains the elements of Galois Field
    tmp,i;
  // sequence of m ones
  int mask = (1<<m) -1;  // 1(m) sequence of m ones

  // Allocation and initialization of the tables of the Galois Field
  *powOfAlpha = (int *)calloc((1<<m)-2, sizeof(int));
  *indexAlpha = (int *)calloc((1<<m)-1, sizeof(int));

  (*powOfAlpha)[0] = 1;
  (*indexAlpha)[0] = - 1; // conventionally we set -1
  (*indexAlpha)[1] = 0;

  for (i = 0, reg = 1; i < (1<<m)-2; i++)
  {
      tmp = (int)(reg & (1<<(m-1))); // Get the MSB
            reg <<= 1;   // Shift
            if( tmp) {
        reg ^= poly;
        reg &= mask;
      }
      // Step-by-step writing of the tables
      (*powOfAlpha)[i+1] = (int) reg;
      (*indexAlpha)[(int)reg] = i+1;
    }

}
```

At each clock cycle (evolution of the shift-register state) two tables are written down:

1. Each α^i for $i = 1, \ldots, 2^{16} - 2$ is stored in the `powOfAlpha[i]` vector/table.

2. Inverse of the `powOfAlpha[i]` vector/table is saved in the vetor/table `indexAlpha[i]` which, given the α^i value (in decimal notation), provides the correspondent exponent of primitive element α. In practice, this vector/table expresses a base α logarithm (i.e. $\log_\alpha$ operator).

Usefulness of these two vector/tables will be made clear later on.

6.4 Decoding BCH

Algebraic decoding of a binary BCH code consists of the following steps:

1. Computation of syndrome (this is implemented by `errordetection` function).

2. Determination of an *error locator polynomial*, whose roots provide an indication of where the errors are. There are two ways of finding the locator polynomial: Peterson's algorithm and the Berlekamp-Massey algorithm (used in our simulations). In addition, there are techniques based on Galois-field Fourier transforms.

3. Finding the roots of the error locator polynomial. Usually, this is done using the *Chien search*, which is an exhaustive search over all elements in the field.

6.4.1 Error Detection

Before trying to decode a BCH codeword, a preliminary error detection must be accomplished. Parity check (3.12) evaluation in the $2t$ roots of polynomial generator provides the code syndrome

$$\mathbf{S} = S_1 = c(\alpha), S_2 = c(\alpha^2), \ldots, S_{2t} = c(\alpha^{2t}) \tag{6.1}$$

If the codeword received has not been corrupted by noise, then the syndrome is null since $g(\alpha) = g(\alpha^2) = \ldots = g(\alpha^{2t})$; otherwise it is not and then the decoding should be accomplished to exploit redundancy introduced and try to correct errors occurred. Values in (6.1) lie over $GF(2^{16})$ and all the decoding operations have to be carried out in the Galois field where the roots are.

```
bool error_detection(int *pow, int *index, int t)
{
  bool syn = false;
  for(int i = 0; i < t*2; i++)
  {
    S[i] = 0;
    for(int j = 0; j < MAXN; j++){
      if(codeword[j])
        S[i] ^= pow[((i+1)*j)%MAXN];
    }
    if((S[i] = index[S[i]]) != -1)
      syn = true;
```

```
  }
  return syn;
}
```

To evaluate the $2t$ equations relevant to the syndrome, the tables/vectors precomputed (because roots are always in $GF(2^{16})$, given that BCH code is *shortened*) are necessary to compute the $2t$ syndromes. In fact, the function `errordetection()` takes as parameters `powOfAlpha[], indexAlpha[]` vectors. They are useful to

- compute products among elements of $GF(2^{16})$ expressed in exponential form (recall that each element in Galois fields can be written as any power of any primitive element), exploiting exponential properties which still hold in finite fields;
- find the correspondent power of the primitive element which represents, for example, the sum between two elements of GF (e.g. $\alpha^2 + \alpha^{11}$).

If any error is detected (i.e. `syn` is `true`) then the codeword corrupted can be corrected using the Berlekamp-Massey algorithm.

6.4.2 Berlekamp-Massey Algorithm

Before giving a brief description of the algorithm, let us introduce some notation and mathematical concept. Returning to (6.1), suppose that $\mathbf{r}$, the received vector, contains ν errors in locations $i_1, i_2, \ldots, i_\nu$ so that each element of $\mathbf{S}$ can be rewritten as

$$S_j = \sum_{l=1}^{\nu} (\alpha^j)^{i_l} = \sum_{l=1}^{\nu} (\alpha^{i_l})^j \tag{6.2}$$

Letting $X_l = \alpha^{i_l}$, we obtain the following $2t$ equations

$$S_j = \sum_{l=1}^{\nu} {X_l}^j \qquad j = 1, 2, \ldots, 2t \tag{6.3}$$

in the ν unknown error locators. In principle this set of nonlinear equations could be solved by an exhaustive search, but this would be computationally expensive or intractable. To overcome this problem, a new polynomial is introduced, the *error locator polynomial* which casts the problem in a different, and more tractable, setting.

The error locator polynomial is defined as

$$\Lambda(x) = \prod_{l=1}^{\nu} (1 - X_l x) = \Lambda_\nu x^\nu + \ldots + \Lambda_1 x + \Lambda_0 \tag{6.4}$$

where $\Lambda_0 = 1$. In practice, the roots of (6.4) are at the reciprocals of the error locators.

It is possible to demonstrate (for this proof and further information see [15]) that there exist a linear feedback shift register relationship between the syndromes and the coefficients of the error locator polynomial. In practice, we can write

$$S_j = -\sum_{i=1}^{\nu} \Lambda_i S_{j-i} \qquad j = \nu+1, \nu+2, \ldots, 2t \tag{6.5}$$

This formula describes the output of a linear feedback shift register (LFSR) with coefficients $\Lambda_1, \Lambda_2, \ldots, \Lambda_\nu$. From this point of view, the decoding problem consists of finding Λ_j coefficients in such a way that the LFSR generates the known sequence of syndromes $S_1, S_2, \ldots, S_{2t}$.

In the Berlekamp-Massey algorithm, we build the LFRS that produces the entire sequence $\{S_1, S_2, \ldots, S_{2t}\}$ by successively modifying an existing LFSR, if necessary to produce increasingly longer sequences. Clearly, we start with an LFRS that could produce S_1. We determine if that LFRS could also produce the sequence $\{S_1, S_2\}$; if it can, then no modifications are necessary. If the sequence cannot be produced using the current LFSR configuration, we determine a new LFRS that can produce the longer sequence. Proceeding inductively in this way, we start from an LFSR capable of producing the sequence $\{S_1, S_2, \ldots, S_{k-1}\}$ and modify it, if necessary, so that it can also produce the sequence $\{S_1, S_2, \ldots, S_k\}$. At each stage, the modification to the LFRS are accomplished so that the LFSR is the shortest possible. By this means, after completion of the algorithm an LFSR has been found that is able to produce $\{S_1, S_2, \ldots, S_{2t}\}$ and its coefficients correspond to the error locator polynomial $\Lambda(x)$ of smallest degree.

Since we build up the LFSR using information from prior computations, we need a notation to represent the $\Lambda(x)$ used at different stages of algorithm. Let L_k denote the length of LFSR produced at stage k of the algorithm. Let

$$\Lambda^{[k]}(x) = 1 + \Lambda_1^{[k]} x + \cdots + \Lambda_{L_k}^{[k]} x^{L_k} \tag{6.6}$$

be the *connection polynomial* at stage k, indicating the connections for the LFSR capable of producing the output sequence $\{S_1, S_2, \ldots, S_k\}$. That is,

$$S_j = -\sum_{i=1}^{L_k} \Lambda_i^{[k]} S_{j-i} \qquad j = L_k+1, L_k+2, \ldots, k \tag{6.7}$$

At some intermediate step, suppose we have a connection polynomial $\Lambda^{[k-1]}(x)$ of length L_{k-1} that produces $\{S_1, S_2, \ldots, S_{k-1}\}$ for some $k-1 < 2t$. We check if this connection polynomial also produces S_k by computing the output

$$\hat{S}_k = -\sum_{i=1}^{L_k-1} \Lambda_i^{[k-1]} S_{k-i} \tag{6.8}$$

If $\hat{S}_k$ is equal to S_k, then there is no need to update the LFSR, so $\Lambda^{[k]}(x) = \Lambda^{[k-1]}(x)$ and $L_k = L_{k-1}$. Otherwise, there is some nonzero *discrepancy* associated with $\Lambda^{[k-1]}(x)$,

$$d_k = S_k - \hat{S}_k = S_k + \sum_{i=1}^{L_k-1} \Lambda_i^{[k-1]} S_{k-i} = \sum_{i=0}^{L_k-1} \Lambda_i^{[k-1]} S_{k-i} \tag{6.9}$$

In this case we can update the connection polynomial using the formula

$$\Lambda^{[k]}(x) = \Lambda^{[k-1]}(x) + Ax^l \Lambda^{[m-1]}(x) \tag{6.10}$$

where A is some element in the field, l is an integer, and $\Lambda^{[m-1]}(x)$ is one of the prior connection polynomials produced by our process associated with nonzero discrepancy d_m. Using this new connection polynomial, we compute the new discrepancy, denoted by $d_k{}'$, as

$$d_k{}' = \sum_{i=0}^{L_k} \Lambda_i^{[k]} S_{k-i} \tag{6.11}$$

$$= \sum_{i=0}^{L_k-1} \Lambda_i^{[k-1]} S_{k-i} + A \sum_{i=0}^{L_m-1} \Lambda_i^{[m-1]} S_{k-i-l} \tag{6.12}$$

Now, let $l = k - m$. Then, by comparison with the definition of the discrepancy in (6.9), the second summation gives

$$A \sum_{i=0}^{L_m-1} \Lambda_i^{[m-1]} S_{m-i} = Ad_m \tag{6.13}$$

Thus, if we choose $A = -d_m^{-1} d_k$, then the summation in (6.12) gives

$$d_k{}' = d_k - d_m^{-1} d_k d_m = 0 \tag{6.14}$$

So the new connection polynomial produces the sequence $\{S_1, S_2, \ldots, S_k\}$ with no discrepancy. We do not investigate here on how to find the shortest LFSR reproducing syndromes with no discrepancy. We assume that below algorithm could do that. For a complete treatment on LFSR length in Massey algorithm see [15].

Algorithm shown below use the following variables

- Vectors/tables `pow`, `index` to simplify multiplications among the elements of the Galois field
- Vector `c[]`, which indicates the current connection polynomial, that is

  $$c(x) = \Lambda^{[k]}(x)$$

  Vector `p[]`, which indicates a previous connection polynomial, in formula

  $$p(x) = \Lambda^{[m-1]}(x)$$

- Variable `d`, which indicates the discrepancy computed at the current time; variable `dm`, which however indicates the discrepancy computed with $p(x)$ connection polynomial, i.e., at any previous time
- An auxiliary vector `T[]` to be used during the update cycle (with length change) of vector `c[]`
- Variable `l` represents the amount of shift in update, namely $l = k - m$; `L` contains the current length of the LFSR

Let us start to briefly describe the set of operations carried out by the `BerlMass()` C function. After the initialization step, the following operations are accomplished:

1. Compute discrepancy by $S_k + \sum_{i=i}^{L} c_i S_{k-i}$ calculation. This computation is made by the useful pre-computed tables `pow[], index[]`, and using the ANSI C bit-wise operators.
2. If the result of that calculation is zero, then there is no change in polynomial and `l` turns `l+1`. If discrepancy is nonzero, then there exist two options yet:
   - If the double of the current LFSR length is greater equal than `k`, the step counter, then $c(x)$ is updated, but retains its length.
   - Else $c(x)$ length and values change together. The non-updated $c(x)$ is saved into $p(x)$ and so the discrepancy associated to is stored into `dm`. The amount of shift in update turns 1.

After $2t$ cycles, $c(x)$ contains the coefficients of error locator polynomial $\Lambda(x)$.

```
void BerlMass(//int *S, // array of syndrome in exponential notation
        int t2, // length of array S
        int *pow,
        int *index)

{
  int k,L,l,i;
  int d, dm, tmp;
  int *T, *c, *p, *lambda,*el;
  /* Allocation and initialization of local variables */
  /* Auto-Regressive-Filter coefficients
          computed at the previous step */
  p = (int*) calloc(t2,sizeof(int));
  /* Auto-Regressive-Filter coefficients
       computed at the current step */
  c = (int*) calloc(t2,sizeof(int));
  /* Temporary array */
  T = (int*) calloc(t2,sizeof(int));
  /* error location array (found by Chien Search) */
  el = (int*) calloc(t2,sizeof(int));
  /* Error polynomial locator */
  lambda = (int*) calloc(t2,sizeof(int));


  /* Initialization step */
  c[0] = 1;
  p[0] = 1;
  L = 0;
  l = 1;
  dm = 1;

    /* Berlekamp-Massey Algorithm */
```

```
  for (k = 0; k < t2; k++)
  {
    /* Discrepancy computation */
    if(S[k] == -1)
      d = 0;
    else
      d = pow[S[k]];
    for(i = 1; i <= L;i++)
      if(S[k-i] >= 0 && c[i] > 0)
      d ^= pow[(index[c[i]]+ S[k-i])%MAXN];
      /*Multiplication of alpha power */
    if( d == 0)
    {
      l++;
    }
    else
    {
      if(2*L > k)
      {
        for( i = l; i <t2; i++)
        {
          if(p[i-l] != 0)
            c[i] ^= pow[(index[d]-index[dm]+index[p[i-l]]+MAXN)%MAXN];
        }
        l++;
      }
      else
      {
        for( i = 0; i < t2; i++)
          T[i] = c[i];
        for( i = l; i <t2; i++)
        {
          if(p[i-l] != 0)
            c[i] ^= pow[(index[d]-index[dm]+index[p[i-l]]+MAXN)%MAXN];
        }
        L = k-L+1;
        for( i = 0; i < t2; i++)
          p[i] = T[i];
        dm = d;
        l = 1;
      }

    }
  }



    /* Storing of error locator polynomial coefficient */
  for(i = 0; i <=L; i++)
    lambda[i] = index[c[i]];

    /* Chien search */
    /* Roots searching */

  int j;
  k = 0;
  for(i = 0; i < MAXN; i++)
  {
    for(j = 1, tmp = 0; j <=L; j++)
      tmp ^= pow[(lambda[j]+i*j)%MAXN];
    if (tmp == 1)
      // roots inversion give the error locations
      el[k++] = (MAXN-i)%MAXN;

  }
  bool success = true;
  fprintf(o3,"\nPosition of errors detected:\n");
  for(i = 0; i <k; i++) {
    if(el[i] != err[i]) {success=false;}
    fprintf(o3,"%d\t",el[i]);
  }
  if(success) {fprintf(o3,"\nSuccessful decoding!");
  fprintf(stdout,"\nSuccessful decoding!\n----------------------\n");};

fprintf(o3,"\n\n----------------------
------------------------------------------------");


}
```

6.4.3 Chien Search

Having now the error locator, its root must be found in our field of interest ($GF(2^{16})$. Since the search must be accomplished in a finite field, we can examine every element of the field to determine if it is a root. Thus polynomial $\Lambda(x)$ can be evaluated at each nonzero element of the field in succession: $x = 1, x = \alpha, x = \alpha^2, \ldots, x = \alpha^{2^{16}-2}$.

This can be easily implemented by two nested `for` cycles: the outer scans all the elements in GF, the inner evaluate the polynomial performing the above substitution and accumulating (step-by-step) in a `tmp` temporary variable its result. Clearly, if the overall result of this calculation gives 1 (in index form), then a root has been found. Supposing, for example, to have found α^{12} as root. Its inversion can be easily accomplished using some elementary properties of finite fields: expression α^{-12} can be thought as $\alpha^{-12}\alpha^{2^{16}-1} = \alpha^{2^{16}-13}$.

If the roots found are distinct and all lie in the reference field, then we use these to determine the error locations. If they are not distinct or lie in the wrong field, then the received word is not within distance t of any codeword. The correspondent error pattern is said to be an uncorrectable error pattern. An uncorrectable error pattern results in a decoder failure.

6.5 Software Robustness and Validation

The first proof of software functioning consists in a set of simulations to validate the package developed in C programming language and thus support VHDL designers in synthesization of the BCH encoder architecture. To this latter specific purpose, it was convenient setting up a package which can be used to simulate not only the correct functioning of the envisaged parallel encoder but, above all, his capacity of correcting t errors on a operating mode basis.

This has been accomplished interconnecting each block of an ideal communication chain composed, in order, by:

A Pseudo-Noise Source: a LFSR with optimum taps, i.e., capable of generate sequences of maximum period;

BCH Encoder/s which emulates each kind of architectures and algorithms already discussed. The type of algorithm and relevant architecture to emulate can be selected by the user before starting simulation cycle.

An Error Pattern Generator: without any loss of generality, the macro-block channel plus modem has been substituted by a pseudo-random error pattern generator. It generates an IID (Independent Identically Distributed) stochastic process whose samples, representing the errors occurred during the frame

transmission, are distributed as an uniform p.d.f. In practice, we shall simulate an hard detection/corretion of errors.

An Error Detection Block and Syndrome Calculator: this block have the simple task of computing syndromes associated to a received codeword. Clearly, if any error is occurred then the codeword must be processed by decoder to try correcting errors. Otherwise, no codewords are needed being corrected.

A Berlekamp Massey Decoder: when the number of errors is less or equal to t, Berlekamp Massey algorithm and, in turn, the Chien search of roots can correct the errors occurred during transmissions; in the other cases the decoder state its failure in decoding.

In order to verify the functionality of all the algorithms, the positions of error found at the decoder size is compared with the positions added by the error pattern generation function. In case of any mismatch with the added error positions between decoder and error generator, a decoding failure is stated. The result of each simulation cycle can be saved onto a file, which can be indicated by the user.

6.5.1 Error Pattern Generation

Positions of errors are determined by using a C function (`uniform01()`) generating a uniform distributed r.v. between $[0, 1[$, namely X; then

$$Y = \lfloor n_{\text{bch}} \cdot X \rfloor$$

represent the random error location, distributed uniformly between $[0, n_{\text{bch}} - 1]$.

6.5.2 The C Code

This section provides the code used in the simulation campaign (the implementation of `uniform01()` will be omitted).

```
#include <stdlib.h>
#include <conio.h>
#include <iostream.h>
#include <time.h>
#include <dos.h>
#include <stdio.h>
#include <string.h>
#include <fstream.h>
#include <math.h>


#define MAXR 192 // max r bits
#define P 8      // degree of parallelism
#define MAXN ((1<<16)-1)  // primitive code length
#define MAXT 12          // Max corrective capacity
#define DRIFT 0 // adding extra errors


#ifndef TESTENC
#define TESTDEC
#define SERIAL
```

```
#endif



/////////////////////////// MACRO ////////////////////////////////
// It returns corrective capacity of the code with (n,k) given //
//////////////////////////////////////////////////////////////////

#define t(n,k)  ( ((n)-(k)) / (16) )


/*************************************************************************/
/********************** Global Variable  *********************************/
/*************************************************************************/

int codeword[MAXN],
  message[MAXN]; // information bits

int Ap_n[MAXR][MAXR]; // (n-k) rows, (n-k) col
int Ap_k[MAXR][MAXR]; // 192 rows, 192 col
int C[MAXR][P];       // 192 rows, 8 col
int S[(MAXT + DRIFT)*2];           // Syndrome vector
int err[MAXT+DRIFT];            // array of random error location
FILE *o3;

/*************************************************************************/
/********************** PN bit source ************************************/
/*************************************************************************/

int lfsr(unsigned long int *seed)
{
  int b,c;

  b = ( ((*seed) & (1 << 31) ) >> 31 ) ;

  c =   ((*seed) & 1) ^ ( ((*seed) & (1 << 1)) >> 1 ) ^ ( ((*seed) & (1 << 21)) >> 21 ) ^ b ;

  (*seed) = ((*seed) << 1) | c;

  return(b);
}

/*************************************************************************/
/********************** Message generator ********************************/
/*************************************************************************/

void message_gen(int n,int k, unsigned long int  *seed)
{
  int i;
    // Message bits pseudo random generation
  for (i=n-1;i>=n-k;i--)
    message[i] = lfsr(seed);
  // Zero padding
  for(i = 0; i < n-k; i++)
    message[i] = 0;
}

/*************************************************************************/
/********************** Polynomial Generators ****************************/
/*************************************************************************/
// Note: only used by algorithm emulating the serial architecture (n-clock cycles)

const unsigned int gen12[] =
{1,1,1,0,0,1,1,1,1,0,1,0,1,0,1,0,0,1,0,0,0,0,0,0,1,1,0,0,1,1,0,
 1,1,1,0,1,1,1,1,1,0,1,0,0,0,0,1,1,1,1,0,0,0,1,0,1,1,0,0,0,0,0,0,
 1,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,1,0,1,1,0,0,0,0,1,1,1,0,1,1,
 0,0,0,1,1,0,1,1,0,0,1,1,0,1,0,0,1,1,1,1,0,0,1,1,0,0,0,0,1,0,1,0,
 0,0,1,1,1,0,0,0,1,0,0,0,1,0,1,0,0,0,1,1,1,0,1,0,0,0,1,0,0,0,0,1,
 1,1,0,0,0,0,0,1,0,1,1,1,0,0,0,0,0,1,1,0,0,1,0,0,0,1,1,1,0,0,1,0,
 1};
// i.e. gen(x) = a_0*x^0 + a_1*x^1 + ... + a_(r-1)*x^(r-1) + a_r*x^r

const unsigned int gen10[] =
{1,0,0,0,1,0,0,1,1,0,1,0,0,1,1,0,1,1,1,0,1,1,1,0,1,0,0,0,1,1,1,0,1,
 1,0,0,0,0,0,0,0,1,1,0,0,0,1,0,0,0,1,0,0,0,1,1,1,1,1,1,0,1,1,1,1,
 1,1,0,0,0,0,0,0,0,1,1,1,0,1,0,1,0,1,0,0,0,1,1,1,1,0,0,1,0,1,0,1,1,0,
 1,1,1,1,1,0,0,0,1,1,0,0,1,1,0,0,0,1,0,1,0,1,0,0,0,0,0,1,1,1,1,1,1,
 1,0,1,1,0,1,1,1,0,0,1,1,0,0,0,0,1,0,1,0,1,0,0,0,0,0,0,0,0,1,1,0,
 1};

const unsigned int gen8[] =
{1,1,0,1,0,1,0,0,0,1,1,0,0,1,1,0,1,0,0,1,1,1,1,1,1,0,0,1,0,0,0,0,0,
```

```
 1,0,1,0,1,1,1,0,1,0,1,1,0,1,1,0,0,0,1,1,1,1,1,1,1,0,0,1,1,0,0,0,
 1,0,1,1,1,1,0,1,1,1,1,0,1,0,0,1,1,1,1,0,0,1,0,0,1,0,0,0,1,1,1,0,
 1,1,1,1,1,0,1,0,1,0,1,0,0,1,0,0,1,1,1,0,0,0,0,0,0,0,1,1,1,0,0,0,
 1};

/*****************************************************************************/
/********************** Serial BCH encoder ***********************************/
/*****************************************************************************/

void BCH_s_enc(int n, int k)
{

  const unsigned int *g;
  int *reg;
  int mem,app,i,j;

/***************  Mode Selection (t-error-correction) **********************/

  switch(n-k) {
  case 192:
    g = gen12;
    reg = (int*)calloc(n-k,sizeof(int));
    break;
  case 160:
    g = gen10;
    reg = (int*)calloc(n-k,sizeof(int));
    break;
  case 128:
    g = gen8;
    reg = (int*)calloc(n-k,sizeof(int));
    break;
  default:
    fprintf(stdout,"Error:simulation aborted!\n");
    fprintf(stdout,"Please insert a n-k couple provided by DVB-S2 FEC\n");
    exit(0);
  }

/********************** Encoding serial algorithm ******************************/
/*************************    n clock ticks ************************************/

/************************ Computing remainder **********************************/

  for (i=n-1; i>=0; i--)
  {
    mem=reg [n-k-1];
    for (j=n-k-2; j>=0; j--)
    {
      app=mem & g[j+1];
      reg[j+1]=reg[j]^app;
    }

    reg[0]= message[i]^(mem & g[0]);

  }

/********************** Codeword in systematic form ****************************/

  for (i=n-1;i>=n-k;i--)
    codeword[i] = message[i];
  for (i=n-k-1; i >=0; i--)
    codeword[i] = reg[i];

  free(reg);
}

/*****************************************************************************/
/********************** Loading matrices routine *****************************/
/*****************************************************************************/

void load_matrices(int n, int k)
{
  FILE *input_Ap_k, *input_C, *input_Ap_n;
  int i,j;

/***********************  Mode Selection (t-error-correction) **********************/

  switch(n-k) {
  case 192:
    input_Ap_k = fopen ("Matrices/ADVBS2_nclk_t12.txt","r");
    input_Ap_n = fopen ("Matrices/ADVBS2_nclk_t12.txt","r");
    input_C = fopen("Matrices/CDVBS2_kclk_t12.txt","r");
    break;
  case 160:
```

```
    input_Ap_k = fopen ("Matrices/ADVBS2_kclk_t10.txt","r");
    input_Ap_n = fopen ("Matrices/ADVBS2_nclk_t10.txt","r");
    input_C = fopen("Matrices/CDVBS2_kclk_t10.txt","r");
    break;
  case 128:
    input_Ap_k = fopen ("Matrices/ADVBS2_kclk_t8.txt","r");
    input_Ap_n = fopen ("Matrices/ADVBS2_nclk_t8.txt","r");
    input_C = fopen("Matrices/CDVBS2_kclk_t8.txt","r");
    break;
  default:
    fprintf(stdout,"Error: loading of matrices failed!\n");
    fprintf(stdout,"Please insert a n-k couple provided by DVB-S2 FEC\n");
    exit(0);
  }


/******************** Loading matrix Ap_n *********************************/
/////////// Note: ONLY this matrix size is variable ////////////////////////


  for ( i=0;i<n-k;i++){
    for ( j=0;j<n-k;j++){
      //fscanf(input_Ap_k,"%d\t",&(Ap_k[i][j]));
      fscanf(input_Ap_n,"%d\t",&(Ap_n[i][j]));
      //power_A[i][j] = load_i;
    }
    //fscanf(input_Ap_k,"\n");
    fscanf(input_Ap_n,"\n");
  }

/******************** Loading matrix Ap_k *********************************/

  for ( i=0;i<MAXR;i++)
    for ( j=0;j<MAXR;j++)
      fscanf(input_Ap_k,"%d\t", &(Ap_k[i][j]));
    fscanf(input_Ap_k,"\n");



/******************** Loading matrix C ************************************/
  for (i=0;i<MAXR;i++)
  {
    for (j=0;j<P;j++)
    {
      fscanf(input_C,"%d\t",&(C[i][j]));
      //comb_C[i][j] = load_c;
    }

    fscanf (input_C,"\n");
  }


  fclose(input_C);
  fclose(input_Ap_n);
  fclose(input_Ap_k);

}

/*****************************************************************************/
/**********  Combinatorial blocks emulation **********************************/
/*****************************************************************************/

/*****************************************************************************/
/*****************  Input comb network  **************************************/
/*****************************************************************************/

int comb_c(int index, int *input)
{
  int out,f,ind;

  out=0;

  ind=P-1;

  for (f=0; f<P; f++)
  {
    out= out ^ ((C[index][f]) & (input[f]));
    ind--;
  }

  return(out);
```

```
}

/*************************************************************************/
/*****************  State comb network  ******************************/
/*************************************************************************/

int comb_n(int index,int r, int *reg_old)
{
  int out,f;

  out=0;

  for (f=0; f<P; f++)

  {
    out=out^(Ap_n[index][r-f-1] & reg_old[r-f-1]);
  }

  return(out);
}

int comb_k(int index, int *reg_old)
{
  int out,f;

  out=0;

  for (f=0; f<P; f++)

  {
    out=out^(Ap_k[index][MAXR-f-1] & reg_old[MAXR-f-1]);
  }

  return(out);
}



/*************************************************************************/
/********************** BCH parellel encoder *****************************/
/********************** n clock ticks        *****************************/
/*************************************************************************/

void BCHnclk_par(int n,int k)
{
  int clock_ticks;
  int *reg, *reg_old;

  int input[P]; // parallel input bits

/***************  Mode Selection (t-error-correction) **********************/

  switch(n-k) {
  case 192:
    reg = (int*)calloc(n-k,sizeof(int));
    reg_old = (int*)calloc(n-k,sizeof(int));
    break;
  case 160:
    reg = (int*)calloc(n-k,sizeof(int));
    reg_old = (int*)calloc(n-k,sizeof(int));
    break;
  case 128:
    reg = (int*)calloc(n-k,sizeof(int));
    reg_old = (int*)calloc(n-k,sizeof(int));
    break;
  default:
    fprintf(stdout,"Error:simulation aborted!\n");
    fprintf(stdout,"Please insert a n-k couple provided by DVB-S2 FEC\n");
    exit(0);
  }
  /// Computation of clock ticks required to compute the remainder after division////
  clock_ticks = n/P;



/************************ Computing remainder *********************************/
  int z=0;

  for (int i=0; i<clock_ticks; i++)
  {
    ///// refresh of state  /////// ///////
    for (int m=0; m<n-k; m++)
      reg_old[m]=reg[m];
```

```
    ////////////////////////////////////////
    /////// loading of parallel input //////
    for (int count=P-1; count>=0; count--)
    {
      z++;
      input[count] = message[n-z];
    }
    ////////////////////////////////////////////
    /// Computing of next values of state /////
    if (clock_ticks >0)
    {

      for (m=0; m<n-k; m++)
      {
        if (m<P)

          reg[m] = input[m]^comb_n(m,n-k,reg_old);

        else

          reg[m] = comb_n(m,n-k,reg_old)^reg_old[m-P];

      }
    }
    ////////////////////////////////////////////
  }
/*****************************************************************************/

/********************** Codeword in systematic form ***************************/

  for (i=n-1; i>n-k-1; i--)
    codeword[i] = message[i];

  for (i=n-k-1; i>=0; i--)
    codeword[i] =  reg[i];

}

/*****************************************************************************/
/********************** BCH parellel encoder *********************************/
/********************** k clock ticks    *********************************/
/*****************************************************************************/

void BCHkclk_par(int n,int k)
{
  int clock_ticks;
  int *reg, *reg_old;
  int offset,m;

  int input[P]; // parallel input bits

/**************  Mode Selection (t-error-correction) **********************/

  switch(n-k) {
  case 192:
    reg = (int*)calloc(MAXR,sizeof(int));
    reg_old = (int*)calloc(MAXR,sizeof(int));
    offset = MAXR-192;
    break;
  case 160:
    reg = (int*)calloc(MAXR,sizeof(int));
    reg_old = (int*)calloc(MAXR,sizeof(int));
    offset = MAXR - 160;
    break;
  case 128:
    reg = (int*)calloc(MAXR,sizeof(int));
    reg_old = (int*)calloc(MAXR,sizeof(int));
    offset = MAXR-128;
    break;
  default:
    fprintf(stdout,"Error:encoding aborted!\n");
    fprintf(stdout,"Please insert a n-k couple provided by DVB-S2 FEC\n");
    exit(0);
  }
  /// Computation of clock ticks required to compute the remainder after division////
  clock_ticks = k/P;

/************************ Computing remainder ********************************/

  int z=0;
  for (int i=0; i<clock_ticks; i++)
  {
    for (m=0; m < MAXR; m++)
```

```
      reg_old[m]=reg[m];

    for (int count=P-1; count>=0; count--)
    {
      z++;
      input[count] = message[n-z];
    }


    if (clock_ticks>0)
    {

      for (m=0; m< MAXR; m++)
      {
        if (m<P)

          reg[m] = (comb_c(m,input))^(comb_k(m,reg_old));
        else
          reg[m] = (comb_c(m,input))^(comb_k(m,reg_old))^(reg_old[m-(P)]);
      }
    }
  }
/********************** Codeword in systematic form ******************************/

  for (i=n-1; i>n-k-1; i--)
    codeword[i] = message[i];

  for (i=n-k-1; i>=0 ; i--)
    codeword[i] =  reg[i+offset];

  /*  Check values of register
  FILE *de;
  de = fopen("debugkclk.txt","w");
  for(i = MAXR-1; i >=0; i--)
    fprintf(de,"%d\n",reg[i]);
  */

}

/*****************************************************************************/
/********************** Creation of GF(2^m)  ********************************/
/********************** useful tables        ********************************/
/*****************************************************************************/

void gfField(int m, // Base 2 logarithm of cardinality of the Field
       int poly, // primitive polynomial of the Field in decimal form
       int ** powOfAlpha, int ** indexAlpha)
{
  int reg,  // this integer of 32 bits, masked by a sequence of m ones,
        // contains the elements of Galois Field
    tmp,i;
  // sequence of m ones
  int mask = (1<<m) -1;  // 1(m) Bit Masking

  // Allocation and initialization of the tables of the Galois Field
  *powOfAlpha = (int *)calloc((1<<m)-2, sizeof(int));
  *indexAlpha = (int *)calloc((1<<m)-1, sizeof(int));

  (*powOfAlpha)[0] = 1;
  (*indexAlpha)[0] = - 1; // we set -1
  (*indexAlpha)[1] = 0;

  for (i = 0, reg = 1; i < (1<<m)-2; i++)
  {
      tmp = (int)(reg & (1<<(m-1))); // Get the MSB
            reg <<= 1;   // Register shifted
            if( tmp) { //
        reg ^= poly;
        //
        reg &= mask;
      }
      // Step-by-step writing of the tables
      (*powOfAlpha)[i+1] = (int) reg;
      (*indexAlpha)[(int)reg] = i+1;
    }


}


/*****************************************************************************/
/********************** Error detection    ********************************/
/*****************************************************************************/
```

```
bool error_detection(int *pow, int *index, int t)
{

  bool syn = false;
  for(int i = 0; i < t*2; i++)
  {
    S[i] = 0;
    for(int j = 0; j < MAXN; j++){
      if(codeword[j])
        S[i] ^= pow[((i+1)*j)%MAXN];
    }
    if((S[i] = index[S[i]]) != -1)
      syn = true;

  }

  return syn;

}


/*****************************************************************************/
/********************** Error correction    ******************************/
/*****************************************************************************/

void BerlMass(//int *S, // array of syndrome in exponential notation
        int t2, // length of array S
        int *pow,
        int *index)

{
  int k,L,l,i;
  int d, dm, tmp;
  int *T, *c, *p, *lambda,*el;
  // Allocation and initialization
  // Auto-Regressive-Filter coefficients computed at the previous step
  p = (int*) calloc(t2,sizeof(int));
  // Auto-Regressive-Filter coefficients computed at the current step
  c = (int*) calloc(t2,sizeof(int));
  // Temporary array
  T = (int*) calloc(t2,sizeof(int));
  // error location array (found by Chien Search)
  el = (int*) calloc(t2,sizeof(int));
  // Error polynomial locator
  lambda = (int*) calloc(t2,sizeof(int));


  // Inizialization step
  c[0] = 1;
  p[0] = 1;
  L = 0;
  l = 1;
  dm = 1;

/********** Berlekamp-Massey Algorithm ******************/
  for (k = 0; k < t2; k++)
  {
    // Discrepancy computation
    if(S[k] == -1)
      d = 0;
    else
      d = pow[S[k]];
    for(i = 1; i <= L;i++)
      if(S[k-i] >= 0 && c[i] > 0)
      d ^= pow[(index[c[i]]+ S[k-i])%MAXN];
      // exponential rule

    if( d == 0)
    {
      l++;
    }
    else
    {
      if(2*L > k)
      {
        for( i = l; i <t2; i++)
        {
          if(p[i-l] != 0)
            c[i] ^= pow[(index[d]-index[dm]+index[p[i-l]]+MAXN)%MAXN];
        }
        l++;
      }
```

```
      else
      {
        for( i = 0; i < t2; i++)
          T[i] = c[i];
        for( i = l; i <t2; i++)
        {
          if(p[i-l] != 0)
            c[i] ^= pow[(index[d]-index[dm]+index[p[i-l]]+MAXN)%MAXN];
        }
        L = k-L+1;
        for( i = 0; i < t2; i++)
          p[i] = T[i];
        dm = d;
        l = 1;
      }

    }
  }


/********** Storing of error locator polynomial coefficient **********/
  for(i = 0; i <=L; i++)
  {
    // Error storing
    lambda[i] = index[c[i]];

  }

/**************     Chien search    ************************/
/******************   Roots searching  **********************/

  int j;
  k = 0;
  for(i = 0; i < MAXN; i++)
  {
    for(j = 1, tmp = 0; j <=L; j++)
      tmp ^= pow[(lambda[j]+i*j)%MAXN];
    if (tmp == 1)
      // roots inversion give the error locations
      el[k++] = (MAXN-i)%MAXN;

  }
  bool success = true;
  fprintf(o3,"\nPosition of errors detected:\n");
  for(i = 0; i <k; i++) {
    if(el[i] != err[i]) {success=false;}
    fprintf(o3,"%d\t",el[i]);
  }
  if(success) {fprintf(o3,"\nSuccessful decoding!");
  fprintf(stdout,"\nSuccessful decoding!\n----------------------\n");};
  fprintf(o3,"\n\n--------------------------------------");


}


// Random variable uniformly distributed between 0.0 and 1.0
extern double uniform01(long * );

/*************************************************************************/
/********************** Insertion sort  ****************************/
/*************************************************************************/

void elSort(int dim)
{
  int i, j;
  int key;

  for ( i = 1; i < dim; i++)
     {
         key = err[i];
         j = i - 1;
         while (err[j] < key && j >=0) {
       err[j+1] = err[j];
       j--;
    }
    err[j+1] = key;
  }

}
```

```
/*************************************************************************/
/******************        MAIN FUNCTION        **************************/
/*************************************************************************/

int main()
{
  FILE *o1, *o2;
  int *pow, *index;
  int n,k,i,s;
  unsigned long int seed;
  long seed2;
  char *outfile = new char[120];


  //o1 = fopen("outserial.txt","w");
  //o2 = fopen("outpnclk.txt","w");

  n = 57600;  k = 57472;  //
  n = 16200; k=16008; // n = 21600; k = 21408; // n = 43200; k = 43040;

#ifndef SERIAL
  load_matrices(n,k);
#endif
#if defined (TESTDEC)
  sprintf(outfile,"DecTest/outdec_%d_%d.txt",n,k);
  o3 = fopen(outfile,"w");
#endif
  // Galois Field Creation
  gfField(16,
    32+8+4+1,
    &pow,
    &index
    );

#if defined (TESTDEC)
  sprintf(outfile,"DecTest/outdec_%d_%d.txt",n,k);
  o3 = fopen(outfile,"w");
#endif

  /** Simulation Loop **/
  for(s = 0; s <100; s++)
  {

  message_gen(n,k,&seed);
#ifdef SERIAL
  BCH_s_enc(n,k);
#endif


#ifdef NPARALLEL
  BCHnclk_par(n,k);
#endif

#ifdef KPARALLEL
  BCHkclk_par(n,k);
#endif


  fprintf(stdout,"SIM #%d\n",s+1);

#if defined (TESTDEC)
  fprintf(o3,"\nSimulation #%d\nLocation of the pseudo-random errors:\n ",s+1);

  // Random error pattern generator
  for(i = 0; i < t(n,k)+DRIFT; i++){
    // bit flipping
    codeword[err[i] = (int)floor(n*uniform01(&seed2))] ^= 1;
    fprintf(o3,"%d\t",err[i]);
  }
  // Sort of the error locations in decreasing order:
  // it will be useful to check the corrispondence with errors detected
  elSort(t(n,k)+DRIFT);
#endif

  if(error_detection(pow,index,t(n,k)+DRIFT) ) {
    fprintf(stdout,"Errors detected!\nDecoding by Berlekamp-Massey algorithm.....\n");
    fprintf(o3,"\n\nErrors detected!\nDecoding by Berlekamp-Massey algorithm.....\n");
    BerlMass((t(n,k)+DRIFT)*2,pow,index);

  }
  else
    fprintf(stdout,"\n\nNo errors detected!\n-------------------------------\n");
```

```
  }
  return 0;
}
```

Chapter 7

Preliminary Laboratory Test of DVB-S2 TX Section

7.1 Introduction

This chapter presents the results of preliminary tests performed on the TAS-I developed TX section of DVB-S2 system whose VHDL has been synthesized, for preliminary validation and test before the equipment production, onto the Stratix II DSP development board. The above section, in particular, includes the BCH section, subject of some of the previous chapters.

A brief description of the TX Section and its individual functional blocks depicted in Figure 7.1 is given below.

Input Interface This is the flexible block acquiring the source data: as many packets as required are acquired depending on the output symbol rate, modulation and coding rate of the PLFRAME to be generated.

DVB-S2 Formatter As described in chapter 3, this block composes the frame. The DVB-S2 normal FEC-frame has a fixed length of 64800 bits, excluding pilot bits insertion.

Base Band Header Insertion This block inserts a fixed length Baseband Header (BBHEADER) of 10 bytes shall be inserted in front of the DATA FIELD, describing its format (the maximum efficiency loss introduced by the BB-HEADER is 0,25% for $n_{\mathrm{ldpc}} = 64800$ and 1% for $n_{\mathrm{ldpc}} = 16200$, assuming that inner code rate is 1/2. As said in chapter 2, Data field is simply constituted by the information bits; finally, zero-bits shall be appended after this field in order to have a constant length of k_{bch} bits (Padding).

BB Scrambling This block is necessary since the complete BBFRAME shall be randomized. The randomization sequence shall be synchronous with the BBFRAME, starting from the MSB and ending after k_{bch} bits. The randomization is realized by adding the BBFRAME sequence (starting from MSB) to a random sequence generated by a linear shift register and initialized at the beginning of each frame. The scrambling sequence has been stored in a ROM memory; informative bits are processed in parallel by means of a standard parallelization of a linear system.

Physical Layer Scrambling Block Prior to modulation, each PLFRAME, excluding the PLHEADER, has to be randomized for energy dispersal by multiplying the $(I+\mathrm{j}\,Q)$ samples by a complex randomization sequence $(CI+\mathrm{j}\,CQ)$. The randomization sequence is re-initialized at the end of each PLHEADER, besides the PLFRAME duration depends on the modulation selected, thus the randomization sequence length shall be truncated to the current PLFRAME length. The scrambling code sequences are constructed by combining two real m-sequences (generated by means of two generator polynomials of degree 18) into a complex sequence. The resulting sequences thus constitute segments of a set of *Gold sequences*. The physical layer scrambling has been generated by a ROM memory which is read starting from the address 0 at the beginning of each frame.

Encoder Block This functional block provides the following functions:

- CRC-8 (1-Byte Cyclic Redundancy Check) insertion on the incoming packets;
- outer (BCH) and inner (LDPC) encoding, according to the selected coding rate (see Table 3.5);
- bit interleaving, when the selected modulation format is an 8PSK.

CRC-8 Encoder This functional block is necessary only for packetized streams only.

Bit Interleaving Block The direct implementation of the Bit Interleaving suggested by Hughes Network Systems (HNS) in the ETSI standard definition strongly reduces the maximum achievable baud rate; in effect maximum baud rate is f_{ck}/η, where f_{ck} is the operating frequency and η is the modulation efficiency.

Later on an alternative parallel architecture is introduced that doesn't present this limitation.

Our goal has been to carry out a parallel bit interleaving by means of a block that outputs one modulation signal per clock cycle in order to maximize the achievable baud rate. In this case, if f_{ck} [MHz] is the operating frequency, we will have a maximum achievable baud rate equal to f_{ck} [MBaud].

The goal is obtained by means of a parallel architecture, which allows to read η consecutive bits per clock cycle belonging to different symbols; in this way, we provide one symbol per clock cycle.

Mapper Block Mapping into constellation is carried out according to the selected modulation format (Q-PSK or 8-PSK); constellations are stored in a ROM (Read Only Memory) and a 7-bit representation has been used for I and Q components.

Shaping Filter The I and Q data paths at the output of the serial to parallel block are processed by the SRRC (Squared Root Raised Cosine) filter and polyphase interpolator included on the same structure.

This filter is used to provide the interpolation and pulse shaping of the data in order to minimize intersymbol interference. The complex valued coded symbols stream is applied at the input of this sub-unit which performs SRRC pulse shaping by a FIR filter running at rate $3R_s$. The choice of a processing rate equal to $3R_s$ is made considering that the desired sampling frequency f_{SA} is achieved by interpolating the shaped signal; as a consequence an adequate attenuation of the signal spectrum replica, centered around the interpolator input sampling frequency, is required.

Interpolator Section In this specific case a third order Farrow Interpolator block has been used to change the rate at the output of the polyphase filter, depending on the input bit rate selected.

Digital Up Conversion (UPC) Stage This block realizes the up-conversion from the baseband frequency to the intermediate frequency.

DAC Precompensation Filter This block has been designed to compensate the signal distortion (about 4 dB from DC to half the sampling frequency) introduced by the D/A conversion; in order to correct this linear distortion a FIR filter is used, just before the DAC.

This is most for a number of reasons, e.g. the need for complex filter coefficients in case compensation were done at base-band; however, high speed operations are actually desirable: hence, a coefficient simplification is introduced to avoid multiplications and the transpose form of a FIR filter has been used; pipelining can also be used for it to further enhance speed if required.

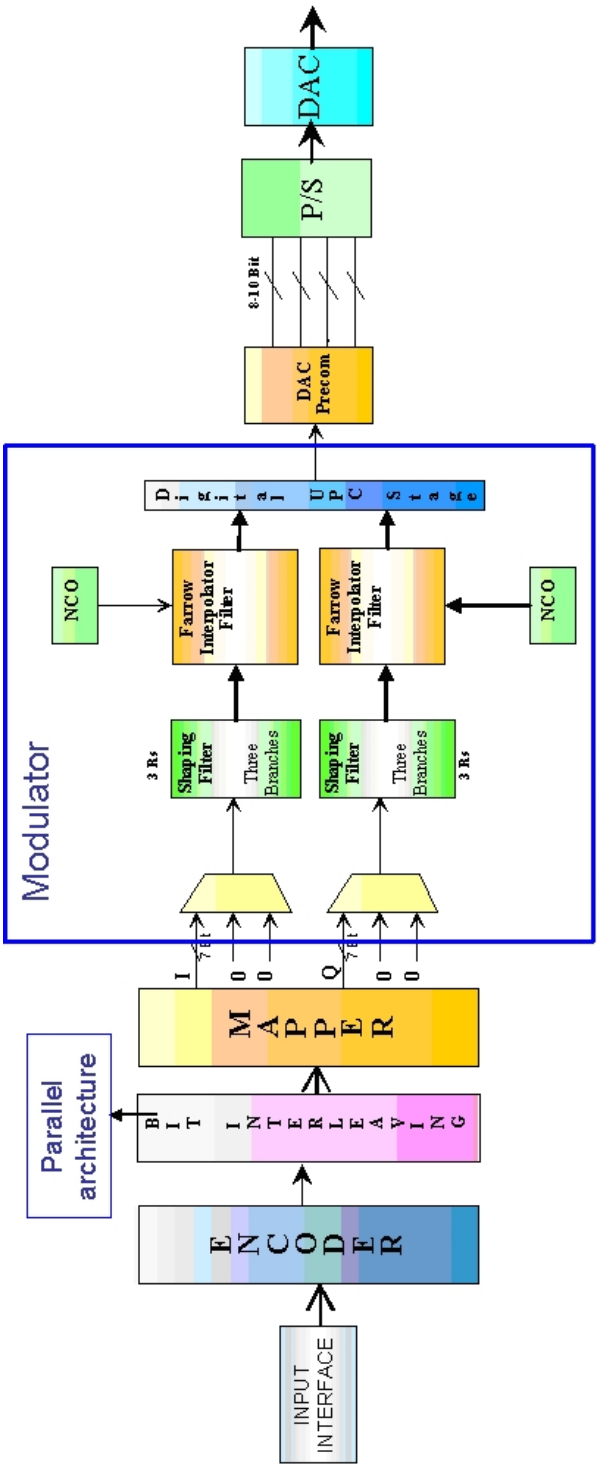


Figure 7.1: TX Section Architecture

7.2 Test Objectives and Setup

Aim of this test campaign is to verify the encoder and modulator design and its performances in terms of constellation implementation quality: this is pursued via a statistical characterization on phase and magnitude error.

All tests have been performed after synthesizing the VHDL-based design onto a Stratix II DSP Development Board, which includes an EP2S180 device. Additionally, this board provides two 14-bit D/A converters with an up to 165-Megasample per second rate and single-ended output.

In order to best evaluate the modulator performances, signal at the output of D/A converter has been routed to an ultra-broadband Vector Signal Analyzer (VSA) which also executes demodulation. The VSA is composed by the association of:

1. a MSO6054A 6000 series oscilloscope (500 MHz bandwidth, 4 Gsa/s);
2. the 89601A vector signal analyzer software, running onto a PC interconnected to the oscilloscope via an Ethernet port.

Test setup equipments is shown in Figure 7.2, Stratix II FPGA hosts the VHDL-based design.

Figure 7.2: Laboratory Test

In particular, sampling rate and IF center frequency have been designed in order to meet the requirements in presence of the HW constraints imposed by both the prototyping board and the anti-image filters available. A parallel to serial converter at the output of modulator has been added to serialize the four parallel data streams at the output of Tx section. Note that the four parallel flows are normally necessary to relax the high speed as well as the relevant criticality in the transfer of data from the TX DSP section to the DAC. The overall transmitter schematic diagram is shown in figure above.

Tx section performances have been evaluated in terms of constellation quality, furthermore, a statistical analysis of magnitude and phase errors has been computed exploiting the dedicated tool available in VSA.

The tool is based on the EVM (Error Vector Magnitude) analysis. EVM is the Root Mean Square (RMS) of the error vectors computed and expressed as a percentage of the square root of the mean power of the ideal signal. The instrument, apart from peak values, can also provide the statistically estimated standard deviation of the EVM and of the phase error.

The error vector is the magnitude of the vector, at the detected symbol location, which connects the I/Q reference-signal phasor to the I/Q measured-signal phasor, and is computed as follows (for a graphical representation refer to the phasor diagram shown below):

$$\mathrm{EVM}[n] = \sqrt{I_{\mathrm{err}}[n]^2 + Q_{\mathrm{err}}[n]^2} \tag{7.1}$$

where the variable n indicates the discrete symbol time and obviously the error in phase is $I_{\mathrm{err}} = I_{\mathrm{Ref}} - I_{\mathrm{Meas}}$, that is, the reference I component is subtracted to the measured component. In a similar way the component of error in quadrature can be measured as follow: $Q_{\mathrm{err}} = Q_{\mathrm{Ref}} - Q_{\mathrm{Meas}}$.

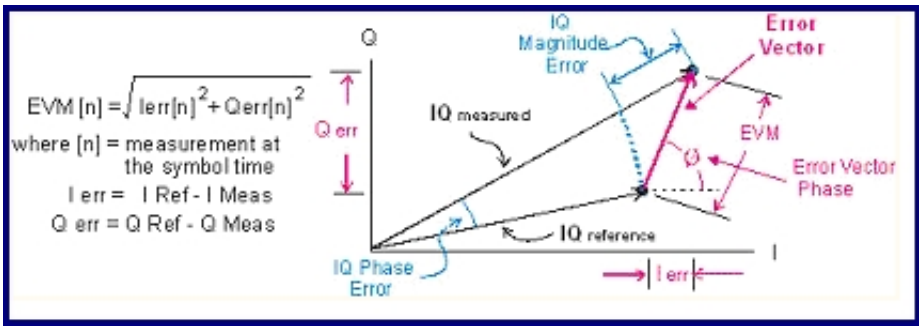


Figure 7.3: Error vector magnitude computation

In addition, magnitude error and phase error (computed as described in Figure 7.4) vs time diagram (expressed in symbols) have been plotted to better evaluate constellation quality.

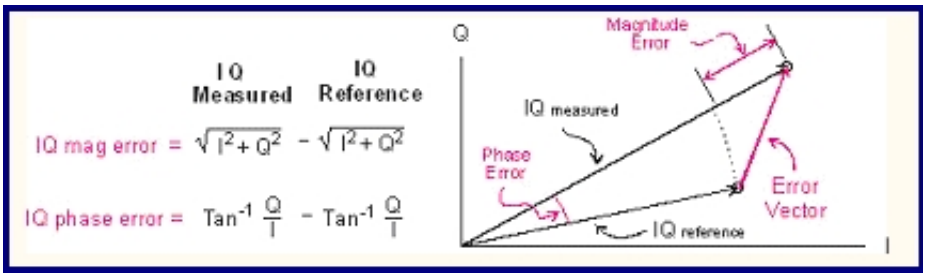


Figure 7.4: Magnitude and phase error computation

7.3 Test Results

In this preliminary test campaign, some measurements on the developed DVB-S2 TX section have been performed. For both 2 MBaud and 30 MBaud symbol rates, demodulated signal constellations have been plotted.

From a statistical point of view, modulator performance have been evaluated computing phase and magnitude error in the way indicated before.

From results of these preliminary test, degradation due to imperfect image rejection and DAC distortion is clear.

7.3.1 2 MBaud — 16 APSK

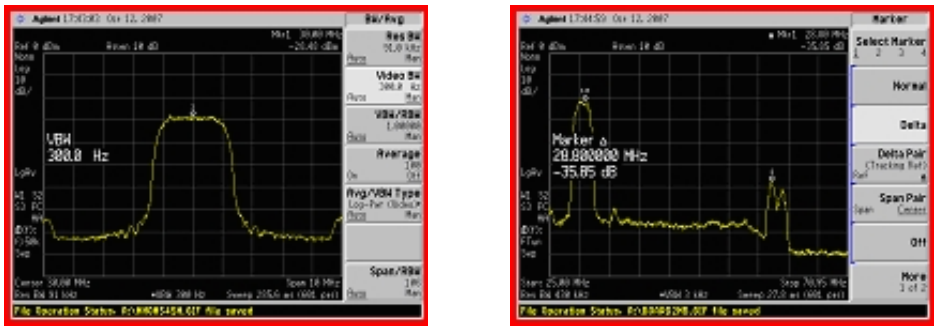
In this test, signal constellation demodulated by VSA has been visually and qualitatively analyzed. In addition, phase and magnitude error have been plotted and quantified via statistical measurements to better analyze the modulation quality.

Figure 7.5: 2MBaud 16 APSK constellation

The statistical analysis of errors leads to the following results:

EVM	2%
Magnitude Error	0,9%
Phase Error	3 deg

A very limited degradation with respect to the ideal constellation is actually evident.

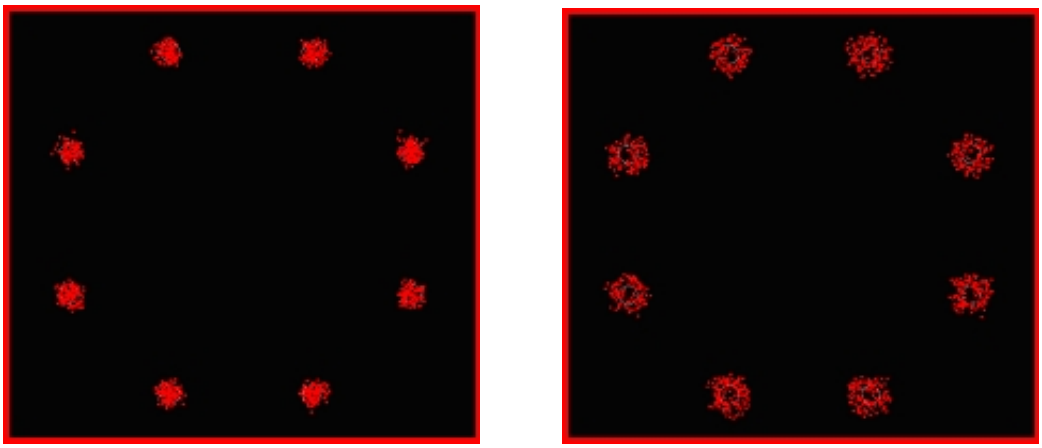


(a) Signal Spectrum (b) Replica Distance

Figure 7.6: 2 MBaud 16-APSK Signal spectra

7.3.2 30 MBaud — 8 PSK

In this test, degradation of modulator performances due to the absence of a DAC pre-compensation filter is clear. In following figures a less scattered plot with pre-compensation filter can be seen.



(a) Signal constellation quality in presence of DAC compensation

(b) Signal constellation quality in absence of DAC compensation

Figure 7.7: 30 MBaud 8-PSK Scatter Plots

Signal spectra comparison in Figure 7.8 shows the distortion introduced by the DAC: the blue curve, as a matter of fact, is the no-precompensated signal spectrum, which results in a very scattered constellation (Figure 7.7b); the yellow one, representing the signal spectrum of the same signal processed by the precompensation filter, results in a better scatter plot.

The statistical analysis of errors leads to the following results:

	Without pre-compensation filter	With pre-compensation filter
EVM	9%	4%
Magnitude Error	6%	2%
Phase Error	3 deg	1,5 deg

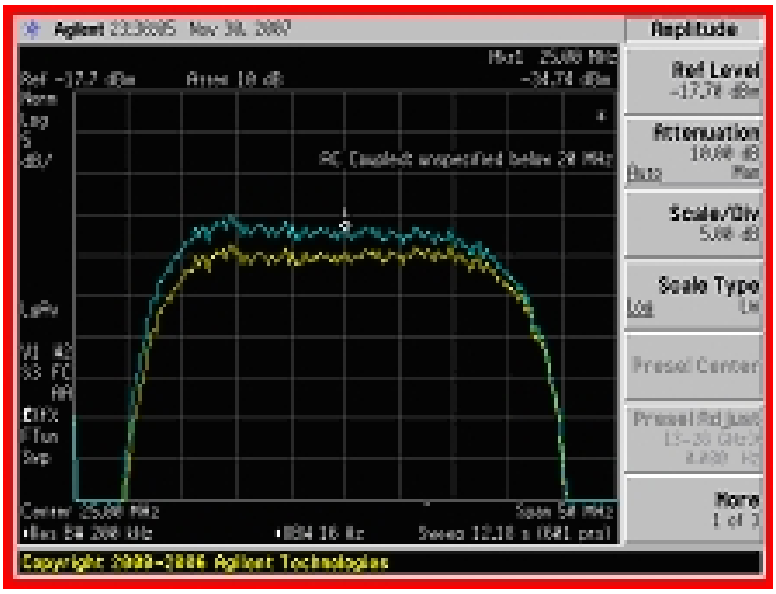
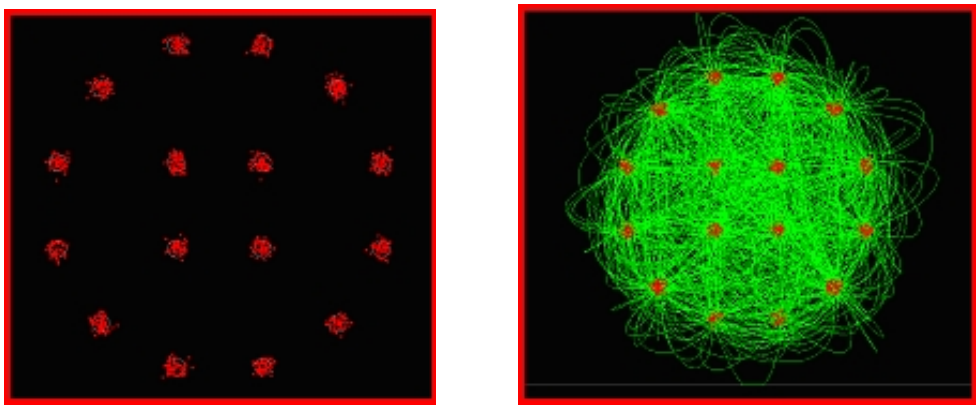


Figure 7.8: Spectra comparison between the spectrum of the signal precompensated (yellow one) and that of the signal non-precompensated (blue one)

7.3.3 30 MBaud — 16 APSK

In this test, signal constellation demodulated by VSA has been visually and qualitatively analyzed. In addition, phase and magnitude error have been plotted and quantified via statistical measurements to better analyze the modulation quality.



(a) Scatter (b) Scatter

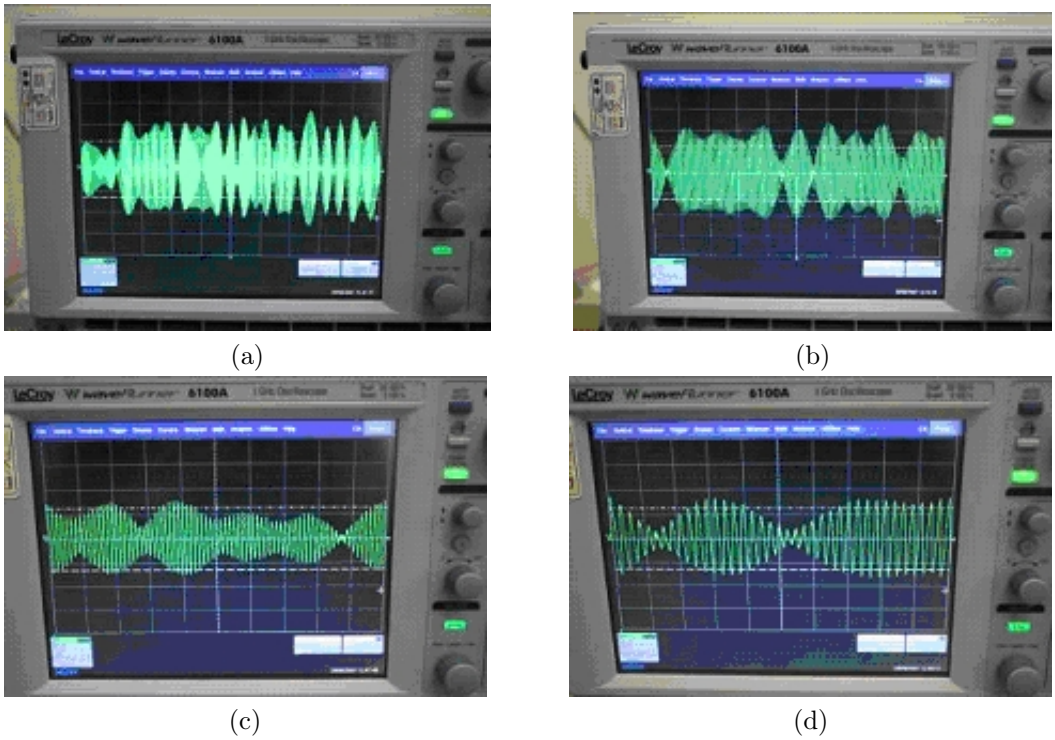
Figure 7.9: 30 MBaud 16-APSK Scatter Plots

The statistical analysis of errors leads to the following results:

EVM	3%
Magnitude Error	1,5%
Phase Error	2 deg

7.4 64-APSK Modulator

The modulator tested by TAS can also provide modulations of higher order with respect to DVB-S2 (e.g. MHOMS modulation schemes). Figure shows a progressive expansion of time-scale representation for a 64-APSK modulation scheme, being this the most innovative modulation, and the four signal envelope level either(points lie on 4 concentric circumferences, that is, each radius represents the amplitude of each level provided).



(a) (b) (c) (d)

Figure 7.10: 64-APSK Signal

Chapter 8

Conclusions

In this thesis we have dealt with a digital communications TX section which can provide a reliable information transmission via satellite while complying with the challenging requirements on performance imposed by DVB-S2 standard.

My most working contributions to the overall project of the DVB-S2 TX section developed by TAS-I have been focused specifically on

- study of the theory underlying the BCH code: Galois fields and cyclic codes theory in order to have the theoretical basis to understand the DVB-S2 code structure and analyze algorithms related to;

- theoretical analysis of DVB-S2 BCH code structure from which we have reach the conclusion that the DVB-S2 outer code (BCH) must be *primitive* and *shortened*;

- study of the BCH encoding procedures and algorithms and its related (serial) architectures;

- modelling and design of a parallel BCH encoder with an high degree of parallelism in order to best match the TX Section (developed by TAS) speed requirements;

- development of a Berlekamp-Massey software decoder in order give a proof of the error correction capacity of the code (encoder); to this purpose we have defined specifical routines aimed to compute arithmetic tables useful in decoding and error detection computations;

- development of a software package (C/C++) in order to verify the functioning of architectures envisaged and give support to VLSI designers in validation of the correspondent VHDL model;

Concerning to the DVB-S2 architecture and overall TX section and thanks to the contribution of the TAS-I Algorithm & Architectures team, this thesis has dealt with:

- identification of the strategic importance of satellite communications in some application scenarios such as Multimedia, Earth observation, radio-localization and navigation, *etc.*;
- review of novel techniques aimed at obtaining top performance by handling modulation and coding schemes jointly;
- assessment of the features of ACM techniques, which allow to better utilization of resources onboard while ensuring an high QoS (Quality of Service);
- design of the DVB-S2 Modem architecture and blocks focusing on concatenated coding (BCH-LDPC) and modulation schemes;
- qualitative description of LDPC encoding/decoding algorithms, and on the latter we have highlighted the importance of the BCH code to counteract against error floors affecting LDPC *iterative decodings*;
- BCH encoding procedures and algorithms and its related (serial) architectures;
- modelling and design of a parallel encoder with an high degree of parallelism in order to best match the TX Section (developed by TAS-I) speed requirements;
- selection between two possible (and functionally equivalent) BCH parallel architectures in order to work in ACM;
- development of a Berlekamp-Massey software decoder in order give a proof of the error correction capacity of the code (encoder); to this purpose we have defined specifical routines aimed to compute arithmetic tables useful in decoding and error detection computations;
- development of a software package (C/C++) in order to verify the functioning of architectures envisaged and give support to VLSI designers in validation of the correspondent VHDL model;
- participation to a preliminary laboratory test and validation of the TAS-I developed DVB-S2 TX section .

Appendix A

Galois (or Finite) Fields

In this appendix we introduce some basic coding concepts related to error correction coding with a particular focus on mathematical tools come in useful through DVB-S2 FEC encoding. Obviously, we do not provide a complete treatment of such a rigorous and abstract mathematical theory. First, because it not so worth for our purposes; second, because there are several books written over Coding Theory where is exhaustively and rigorously explained all that could be necessary. Therefore, for any further information over the next presented topics, the reader could refer, for example, to [15] or [9].

In engineering application it is often necessary finding a reasonable trade-off between system complexity and performance achieved. On the one hand, elementary decoding approach is simpler, but, on the other hand, its computational burden increases exponentially with the length of code. Algebraic decoding represents a valid alternative to overcome such an huge complexity grow.

The key idea concerning the algebraic coding is that of giving to codes some algebraic structure to be exploited for decoding purposes. When length of block become rather great decoding turn out to be somewhat intractable. For example, the hard decision on the minimum distance between any received sequence and codewords requires an computational effort and a big amount of memory, both increasing with the length of code. An other approach is the *standard array decoding*. Unfortunately, this also turns out to be too expensive when the block dimensions become large. In particular, its memory requirements make this approach rather unattractive.

Algebraic coding/decoding is based on a large extent of powerful and elegant algebraic structures called finite fields. A field is essentially a set of elements where addition, subtraction, multiplication and division operations yield always an element within such set. A well known example of a field is $\mathbb{R}$, the infinite field of real numbers. Although the following observation could be rather trivial, the need of such an algebraic structure is straightforward. Finite fields act as the most common

real fields. The unique difference is that the size of the set is finite as, for example, an ensemble of codewords. Note, for example, that any codeword could not be decoded by Berlekamp-Massey algorithm or Galois Field Transform if inverse property was not satisfied.

Although the preliminary concepts over axioms and elementary algebraic structures which lead to rigorous mathematical definition of field properties are even important, the real core of this appendix is the part as to factorization of a 'special polynomial'. This operation provides, together with some elementary knowledge on the Coding Theory, a powerful tool to design and well understand algebraic encoding and decoding algorithms.

A.1 Algebraic Structures: a Glance

Before discussing properties of finite fields, let us define some preliminary algebraic structures. Let us start from the definition of groups. On this sets it is possible to define some (binary) operations between their elements. To be called a group, operations over such a set must satisfy these following algebraic properties.

Definition A.1.1. A binary operation $*$ on a set assigns to each ordered pair of elements of the set (a, b) some element of the set. (This is defined as *closed* binary operation.)

Definition A.1.2. A *group* is a set $\langle G, + \rangle$ together with a binary operation $*$ on G such that:

G1 The operator is *associative*: for any $a, b, c \in G, (a * b) * c = a * (b * c)$.

G2 There is an element $e \in G$ called the *identity element* such that $a * e = e * a$ for all $a \in G$.

G3 For every $a \in G$, there is an element $b \in G$ known as the *inverse* of a such that $a * b = e$ The inverse of a is sometimes denotes as a^{-1} (when the operator $*$ is multiplication-like) or as $-a$ (when the operator $*$ is addition-like).

G4 If also $a * b$ is equal to $b * a$, the group is also an abelian (or commutative) group.

If the commutative property holds then a group is said abelian. It is possible to define an algebraic structure as a group also on a finite set of elements, as well as on those infinite.

Definition A.1.3. If G has a finite number of elements, it is said to be a finite group. Furthermore, such number is called the order (or the cardinality) of the group.

Let $\langle \mathbb{Z}_n, + \rangle$ denote addition on the numbers $\{0, 1, \ldots, n-1\}$ modulo n. Clearly such set is an abelian group, since, as it is shown in explicative Table A.2 (which can be easily generalized) 0 is the identity element appearing in each column; moreover, the uniqueness of solution proves associative property; furthermore, the addition table or its generalization is symmetric and thus the operation is commutative.

Let us now introduce another algebraic structure, the *ring*, which will be later helpful in the examination of cyclic codes. Differently from groups, *rings* have two binary operation associated with them.

Definition A.1.4. A *ring* (loosely speaking) is a set where addition, subtraction and multiplication are possible. Formally, R is an additive abelian group, together with a multiplication satisfying $ab = ba$, $a(b+c) = ab+ac$, $(ab)c = a(bc)$, and which contains an identity element 1 such that $1 \cdot a = a$.

As also shown in Table A.2, the multiplication under $\mathbb{Z}_n$ does not form a group: there is a presence of zeros in correspondence of columns which denote the nonzero elements (see the next section for the *zero divisor* formal definition). In fact, in a ring, not every element has a multiplicative inverse, whereas, in a *field*, the familiar arithmetic operations that take place in the usual real numbers are all available.

Definition A.1.5. A *field* is a set of objects $\mathbb{F}$ on which the operations of addition and multiplication, subtraction (or additive inverse), and division (or multiplicative inverse) apply in a manner analogous to the way these operation work for real numbers. That is

F1 The elements of $\mathbb{F}$ form an abelian (or commutative) *group* under addition.

F2 The non-zero elements of $\mathbb{F}$ form an abelian *group* under multiplication.

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

Table A.1: Addition table under $\mathbb{Z}_5$

A.2 How Do We Get Galois Fields?

Let us now try to find a relationship between groups and fields. More specifically, starting from the rings with addition and multiplication operations modulo n, we

·	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

Table A.2: Multiplication table under $\mathbb{Z}_6$

wish to comprehend how obtaining a finite field and under which hypothesis. To this purpose, it could be interesting to note that, in some cases, each nonzero element of a ring has a multiplicative inverse and so is a field, while, in other ones, there exist some elements without a multiplicative inverse. We could start to observe that when the ring characteristic is a prime number (as five, see Table A.3), we have a ring with elements forming an abelian group under multiplications. The reasons will be made more clear after this following definition as to elements called as zero divisors.

Definition A.2.1. In a ring R, if $a, b \in R$ with both a and b not equal to zero but $ab = 0$, then a and b are said to be *zero divisors.*

In practice, the presence of zero divisors make a ring impossible to have a multiplicative inverse for each its elements and, above all, have a multiplicative structure closed under multiplicative operations.Therefore, if we have no zero divisors then a finite ring is also a Galois (or finite) field. The following theorem defines under which hypothesis no such elements exist.

Theorem A.2.1 (Todd K. Moon, [15])**.** *The ring $\mathbb{Z}_p$ is a field if and only if p is a prime.*

Without making any mathematical calculations, we can observe (as proof) that if p is a prime its multiples over $\mathbb{Z}_p$ cannot exist and, therefore, neither zero divisors.

Besides the finite fields with p prime, there are other finite fields. As the field of complex numbers $\mathbb{C}$ can be constructed as a field extension from the field of real numbers $\mathbb{R}$, these Galois fields are *extension fields* of $\mathbb{Z}_p$. More in detail, extension fields are constructed to create roots of irreducible polynomials that do not have roots in base fields. E.g., the polynomial $f(x) = x^2 + 1$ is irreducible over $\mathbb{R}$, but not over $\mathbb{C}$ where we can write:

$$f(x) = (x + \alpha)(x - \alpha) = (x + \mathrm{j})(x - \mathrm{j}) \tag{A.1}$$

Practically we obtain an extension field from a base field whenever we can find any irreducible polynomial over the base field. Loosely speaking, these polynomials may

be conceived as prime numbers by which (as noticed previously) it is possible to construct any fields.

·	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Table A.3: Multiplication table under $\mathbb{Z}_5$

A.3 A Mathematical Survey

Having shown how to obtain Galois fields, we now lay out some aspects of the theory. We first focus on the additive structure of finite fields, which tells us what size any finite fields can be. Recalling Theorem A.2.1, it follows that there are p elements which behave as a field (i.e., we can define addition and multiplication on them as a field). Thus, $\mathbb{Z}_p$ is a subfield of every Galois fields $GF(q)$. In fact, an assertion can be made:

Theorem A.3.1 (Todd K. Moon, [15])**.** *The order q of every finite fields $GF(q)$ must be a power of a prime.*

By Theorem A.2.1, every finite fields $GF(q)$ has a subfield of prime order p. In fact, it can be shown that $GF(q)$ acts like a vector space over its subfield $GF(p)$. More specifically, if we get

$$a_1\beta_1 + a_2\beta_2 + \ldots + a_m\beta_m \tag{A.2}$$

with coefficients a_i lying over $GF(p)$ and $\beta_i \in GF(q = p^m)$ we can obtain all the distinct elements of $GF(q)$. In other words, each combination of coefficients $\{a_1, a_2, \cdots, a_m\}$ corresponds to a distinct element of $GF(q)$.

This theorem shows that all finite fields have the structure of a vector space of dimension m over a finite field $\mathbb{Z}_p$. For the field $GF(p^m)$, the subfield $GF(p)$ is called the *ground field*.

By definition, each element of a finite field forms a group under multiplication. Let us now try to understand how they behave, giving at the same time some helpful definitions.

Definition A.3.1. Let $\beta \in GF(q)$. The *order* [1] of β, written $\text{ord}(\beta)$ is the smallest positive integer such that $\beta^n = 1$.

Definition A.3.2. An element with order $q-1$ in $GF(q)$ (i.e, it generates all the nonzero elements of the field) is called *primitive element.*

In other words, a primitive element has the highest possible order. Since every elements of the field are representable by that, this 'special element' (or elements) enables the 'power representation' of the field, which much simplify multiplication. In the brackets of this last sentence we have referred to a more than one possible existence of these primitive elements. More specifically, they are $\phi(q-1)$ (i.e., Euler totient function [15]), as following theorem asserts.

Theorem A.3.2 (Todd K. Moon, [15])**.** *There are $\phi(q-1)$ primitive elements in $GF(q)$.*

If $\alpha \in GF(q)$ is primitive, then

$$\{\alpha, \alpha^2, \ldots, \alpha^{q-2}, \alpha^{q-1} = 1\}$$

is the set of nonzero elements of $GF(q)$. But, if $\beta = \alpha^i$ is also primitive then the nonzero elements of the fields are still generated by

$$\{\beta, \beta^2, \ldots, \beta^{q-2}, \beta^{q-1} = 1\}$$

Although these are different generators, these are not different fields, but only different representations.

Since, as we previously said, every field element can be written as $\beta = \alpha^i \in GF(q)$ we have the following.

Theorem A.3.3 (Todd K. Moon, [15])**.** *Every element of the field $GF(q)$ satisfies the equation $x^q - x = 0$. Furthermore, they constitute the entire set of roots of this equation.*

Its proof is trivial: let α a primitive element. Then imposing $\beta = \alpha^i$ yields $(\alpha^i)^{q-1} = (\alpha^{q-1})^i = 1$ which satisfies the equivalent equation $x^{q-1} - 1 = 0$.

A.4 Irreducible and Primitive polynomials

While any irreducible polynomial can be used to construct the extension field, computation in the field is easier if a primitive polynomial is used. We make the following observation:

[1] The nomenclature is slightly misguiding, since we already defined the order of a group.

Theorem A.4.1 (Todd K. Moon, [15]). *Let p a prime. An irreducible mth-degree polynomial $f(x) \in GF(p)[x]$ divides $x^{p^m-1} - 1$.*

Let $GF(q) = GF(p^m)$ be constructed using the irreducible polynomial $f(x)$, where α denotes the root of $f(x)$ in the field: $f(\alpha) = 0$. By Theorem A.3.3, α is a root of $x^{p^m-1} - 1$ in $GF(q)$. Using the division algorithm (i.e., Euclid theorem) write

$$x^{p^m-1} - 1 = g(x)f(x) + r(x) \tag{A.3}$$

where $\deg(r(x)) \leq m$. Evaluating above equation at $x = \alpha$ we obtain

$$0 = 0 + r(\alpha)$$

But the elements of the field $GF(q)$ are represented as polynomials in α of degree less equal to m, so since $r(\alpha) = 0$ it must be that $r(x)$ is a zero polynomial, $r(x) = 0$.

Definition A.4.1. An irreducible polynomial $p(x) \in GF(p)[x]$ of degree m is said to be *primitive* if the smallest positive integer n for which $p(x)$ divides $x^n - 1$ is $n = p^m - 1$.

Theorem A.4.2 (Todd K. Moon, [15]). *The roots of an mth degree primitive polynomial $p(x) \in GF(p)[x]$ are primitive elements in $GF(p^m)$*

In other words, any of the roots can be used to generate the nonzero elements of the field $GF(p^m)$. Furthermore, all the nonzero elements of the field can be generated as power of the roots of the primitive polynomial.

A.5 Factoring $x^n - 1$

To design and understand cyclic codes, it is worth succeeding in factorization of $x^n - 1$ over arbitrary finite fields.

From Theorem A.3.3, we know that every element of $GF(q^m)$ is a root of $x^{q^m-1} - 1$, so

$$x^{q^m-1} - 1 = \prod_{i=0}^{q^m-1} (x - \alpha^i)$$

for a primitive element $\alpha \in GF(q^m)$. To provide a factorization over the field $GF(q)$, the factors $x - \alpha^i$ can be grouped together according to conjugacy classes. First of all, let us describe some concepts related to.

Definition A.5.1. Let $\beta \in GF(q^m)$. The *conjugates* of β with respect to a subfield $GF(q)$ are $\beta, \beta^q, \beta^{q^2}, \beta^{q^3}, ...$

Since the field is finite, the above list must repeat at some point. This is provided by the following.

Definition A.5.2. The smallest positive integer d such that n is divisible by $q^d - 1$ is called *multiplicative order of q modulo n.*

Furthermore, if any $\beta \in GF(q^m)$ have $\text{ord}(\beta) = n$ and d multiplicative order of q modulo n, then $\beta^{q^d} = \beta$. The d elements $\beta, \beta^q, \beta^{q^2}, \ldots, \beta^{q^{d-1}}$ are all distinct. In addition, it can be proved the following.

Theorem A.5.1 (Todd K. Moon, [15])**.** *Let $\beta \in GF(q^m)$ have order n and let d be the multiplicative order of q mod n. Then the coefficients of the polynomial $p(x) = \prod_{i=0}^{d-1}(x - \beta^{q^i})$ are in $GF(q)$. Furthermore, $p(x)$ is irreducible. That is, $p(x)$ is the minimal polynomial for β.*

Conjugacy Class	Minimal Polynomial
$\{0\}$	x
$\{1\}$	$x+1$
$\{\alpha, \alpha^2, \alpha^4\}$	$(x-\alpha)(x-\alpha^2)(x-\alpha^4) = x^3 + x + 1$
$\{\alpha^3, \alpha^6, \alpha^5\}$	$(x-\alpha^3)(x-\alpha^6)(x-\alpha^5) = x^3 + x^2 + 1$

Table A.4: Conjugacy Classes over $GF(2^3)$ with respect to $GF(2)$

Conjugacy Class	Minimal Polynomial
$\{0\}$	x
$\{1\}$	$x+1$
$\{\alpha, \alpha^2, \alpha^4, \alpha^8\}$	$(x-\alpha)(x-\alpha^2)(x-\alpha^4)(x-\alpha^8) = x^4 + x + 1$
$\{\alpha^3, \alpha^6, \alpha^9, \alpha^{12}\}$	$(x-\alpha^3)(x-\alpha^6)(x-\alpha^9)(x-\alpha^{12}) = x^4 + x^3 + x^2 + x + 1$
$\{\alpha^5, \alpha^{10}\}$	$(x-\alpha^5)(x-\alpha^{10}) = x^2 + x + 1$
$\{\alpha^7, \alpha^{11}, \alpha^{13}, \alpha^{14}\}$	$(x-\alpha^7)(x-\alpha^{11})(x-\alpha^{13})(x-\alpha^{14}) = x^4 + x^3 + 1$

Table A.5: Conjugacy Classes over $GF(2^4)$ with respect to $GF(2)$

We now are ready to understand that $x^{q^m-1} - 1$ can be expressed as a product of the minimal polynomials of the nonzero elements. For example the polynomial $x^7 - 1 = x^{2^3-1} - 1$ can be factored over $GF(2)$ as a product of the minimal polynomials shown in Table A.4

$$x^7 - 1 = (x+1)(x^3 + x + 1)(x^3 + x^2 + 1) \tag{A.4}$$

To generalize problem of factoring $x^n - 1$ when $n \neq q^m - 1$, let any element $\beta \neq 1$ such that $\beta^n = 1$ call a nth root of unity. First, we have to determine a field $GF(q^m)$

where nth root of unity can exist. Once found, factorization is accomplished using the minimal polynomials in such field.

Since there are $\phi(n)$ elements of order n in $GF(q^m)$, finding a field $GF(q^m)$ with n roots of unity requires finding an m such that $n|q^m - 1$ (i.e, n must be a divisor of $q^m - 1$). This is usually done by trial and error.

Once any β of order n is found, then *beta* is a root of $x^n - 1$ in that field, and so are the elements $\beta^2, \beta^3, \ldots, \beta^{n-1}$. In other words,

$$x^n - 1 = \prod_{i=0}^{n-1} x - \beta^i \tag{A.5}$$

For example, let now determine an extension field $GF(2^m)$ in which 5th roots of unity exist and express the factorization in term of polynomials in $GF(2)[x]$. Since 5 is the smallest divisors of $2^4 - 1$, the roots of unity are over $GF(16)$. So if we let $\beta = \alpha^3$, α primitive, then $\beta^5 = \alpha^1 5 = 1$. Thus the roots of $x^5 - 1 = x^5 + 1$ in $GF(16)$ are

$$1, \beta, \beta^2, \beta^3, \beta^4$$

which can be expressed in terms of the primitive element α as

$$1, \alpha^3, \alpha^6, \alpha^9, \alpha^{12}$$

Using the minimal polynomials shown in Table A.5 we have

$$x^5 - 1 = (x + 1)(x^4 + x^3 + x^2 + x + 1)$$

Appendix B

Cyclic Codes

We now deal with *cyclic codes*, which have additional algebraic structure to make encoding and decoding more efficient. In other words, they have some structure that can be easily exploited through encoding/decoding stages. Note that one of the most notorious cyclic codes are the Hamming codes. In fact, from a certain point of view, BCH codes can be conceived as an extension of Hamming codes. This 'parallelism' is well shown in [21], [9].

Cyclic codes are based on polynomial operations. A natural algebraic settings for the operation on polynomials is the algebraic structure of a ring. This appendix presents some elementary concepts related to shift operations and linear block codes, focusing on the algebraic description of cyclic codes. Galois fields and cyclic codes theory together are exploited in the design of a cyclic code with a specified minimum distance. The theoretical and practical connection between algebraic structures and code description shows a way of reducing the computational burden in decoding.

B.1 Shift Operation

We now introduce the mathematical definition of cyclic block codes.

Definition B.1.1. A (n, k) block code $\mathcal{C}$ is said to be cyclic if it is linear and if for every codeword $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$ in $\mathcal{C}$, its right cyclic shift $\mathbf{c}' = (c_{n-1}, c_0, \ldots, c_{n-2})$ is also in $\mathcal{C}$.

The operations of shifting and cyclic shifting can be conveniently represented using polynomials. The vector

$$\mathbf{c} = (c_0, c_1, \ldots, c_{n-1}) \tag{B.1}$$

is represented by the polynomial

$$c(x) = c_0 + c_1 x + \ldots + c_{n-1} x^{n-1} \tag{B.2}$$

using the obvious one-to-one correspondence. A (noncyclic) shift can be represented by polynomial multiplication:

$$xc(x) = c_0x + c_1x^2 + \ldots + c_{n-1}x^n \tag{B.3}$$

However, if c(x) was a hypothetic codeword, the above operation obviously would not yield any codeword.

As described in Appendix A, taking the remainder after division by $x^n - 1$ translates into imposing $x^n = 1$. So that getting $xc(x)$ module $x^n - 1$ yields

$$xc(x) = c_{n-1} + c_0x + c_1x^2 + \ldots + c_{n-2}x^{n-1} \tag{B.4}$$

Could this result be any codeword, or rather, any element of the same set? In other words, this point is that mathematicians call closure under a given operation (or operator). Before dealing with this, we need to introduce another algebraic structure, the *ideals*. It will show that every cyclic codes forms ideals in a *ring of polynomials*.

B.2 Rings of Polynomials and Ideals in Rings

Let R be a ring. A polynomial $f(x)$ of degree n with coefficients in R is

$$f(x) = \sum_{i=0}^{n} a_i x^i \tag{B.5}$$

where $a_n \neq 0$. The symbol x is said an *indeterminate*.

Definition B.2.1. The set of all polynomials with an indeterminate x with coefficients in a ring R, using the usual operations for polynomial addition and multiplication, forms a ring called the *polynomial ring*. It is denoted as $R[x]$.

Generally, polynomial multiplication has not an inverse. For example in the ring of polynomials with real coefficients $\mathbb{R}[x]$, there is no polynomial solutions $f(x)$ to

$$f(x)(x^2 + 5x + 1) = x^3 + 3x + 1$$

Let us now consider the residue class of polynomials rings $GF(p)[x]$ modulo $x^n - 1$, denoted as $GF(p)[x]/(x^n - 1)$. We have already shown (A.2) that a quotient ring is a field if and only if $x^n - 1$ cannot be factored over the base field $GF(p)$ (p is a prime number). But let us return to our main question. In fact, we would like to find a set where every its shifted elements is still within such set. Such an algebraic structure is the following.

Definition B.2.2. Let R be a ring. A nonempty subset $I \subseteq R$ is an *ideal* if satisfies the following conditions:

1. I forms a group under the addition operation in R.
2. For any $a \in I$ and any $r \in R$, $ar \in I$.

It seems that a cyclic code form an ideal in a ring of polynomials. Furthermore, for cyclic codes the ideals are principal, as defined by the following.

Definition B.2.3. An ideal I in a ring R is said to be *principal* if there exist some $g \in I$ such that every element $a \in I$ can be expressed as a product $a = mg$ for some $m \in R$.For a principal ideal, such an element g is called the *generator element*. The ideal generated by g is denoted as $\langle g \rangle$:

$$\langle g \rangle = \{hg : h \in R\}$$

Theorem B.2.1 (Todd K. Moon, [15])**.** *Let I be an ideal in $\mathbb{F}_q/(x^n - 1)$. Then*

1. *There is a unique monic polynomial $g(x) \in I$ of minimal degree.*
2. *I is principal with generator $g(x)$.*
3. *$g(x)$ divides $(x^n - 1)$ in $\mathbb{F}_q[x]$.*

For example, it can be shown that in $GF(2)[x]$

$$x^7 - 1 = (x + 1)(x^3 + x + 1)(x^3 + x^2 + 1)$$

In the ring $GF(2)[x]/(x^7 - 1)$, there are ideals corresponding to the different factorization of $x^7 + 1$, so there are the following nontrivial ideals

$$\begin{array}{ccc} \langle x + 1 \rangle & \langle x^3 + x + 1 \rangle & \langle x^3 + x^2 + 1 \rangle \\ \langle (x + 1)(x^3 + x + 1) \rangle & \langle (x + 1)(x^3 + x^2 + 1) \rangle & \langle (x^3 + x^2 + 1)(x^3 + x + 1) \rangle \end{array}$$

B.3 Algebraic Description

As said in Section B.1, cyclic shifting of a polynomial $c(x)$ is represented by $xc(x)$ modulo $x^n - 1$ operation. Taking into account Theorem B.2.1 and what said as far, we can therefore reach the following conclusions as to cyclic codes:

- A (n, k) cyclic code has a unique minimal monic polynomial

$$g(x) = g_0 + g_1 x + \ldots + g_{n-k} x^{n-k} \tag{B.6}$$

  It, which is also the generator of ideal, is called the *generator polynomial* for the code, and its degree represents the redundancy $r = n - k$ of the code.

- Every codeword can be expressed as a multiple of the generator

$$c(x) = m(x)g(x) \tag{B.7}$$

Obviously, the degree of message polynomial $m(x)$ is less than k, so the dimension of the code is k. In other words, there are k independently selectable coefficients in $m(x)$.

Generalization for the systematic coding is straightforward. In fact, by Euclid Theorem application we have

$$m(x)x^{n-k} = q(x)g(x) + r(x) \Rightarrow m(x)x^{n-k} - r(x) = q(x)g(x) \tag{B.8}$$

and therefore if we conceive a codeword as $m(x)x^{n-k} - r(x)$ where $r(x)$ represents the remainder of $m(x)$ after division by $g(x)$, then we obtain an equivalent representation of cyclic code. This is called in literature systematic form and it has the following vector correspondence

$$(-d_0, -d_1, \ldots, -d_{n-k-1}, m_0, m_1, \ldots, m_{k-1}) \leftrightarrow x^{n-k}m(x) - r(x) \tag{B.9}$$

Loosely speaking, this different encoding process has the unique effect of changing one-to-one mapping between messages and correspondent codewords.

- Furthermore, the generator $g(x)$ is a factor of $x^{n-1} - 1$ in $GF(q)[x]$.

Bibliography

[1] A. Morello , V. Mignone . Il sistema DVB-S2 di seconda generazione per la trasmissione via satellite a Unicast. *Elettronica e Telecomunicazioni*, (3), December 2003.

[2] Claude Berrou, Alain Glavieux, Punya Thitimajshima. Near Shannon Limit Error – Correcting Coding and Decoding: Turbo Codes. *IEEE*, 1993.

[3] Claude E.Shannon. A Mathematical Theory of Communications. *The Bell System Technical Journal*, 27:379–423, 623–656, July, October 1948.

[4] D. Giancristofaro , M. Fonte , A. Bernardi , I. Martinazzo , M. Livi , R. Novello , A. Ginesi , S. Benedetto , G. Montorsi , M. Luise , L. Giugno . The MHOMS 1 Gbps Modem for Future Earth Observation.

[5] David J.C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[6] European Telecommunications Standard Institute, 650 Route des Lucioles F-06921 Sophia Antipolis Cedex - FRANCE. *EN 302 307 v1.1.1 — Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications*, March 2005.

[7] European Telecommunications Standard Institute, 650 Route des Lucioles F-06921 Sophia Antipolis Cedex - FRANCE. *TR 102 376 v1.1.1 — Digital Video Broadcasting (DVB); User guidelines for the second generation system for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2)*, March 2005.

[8] Ezio Biglieri. Digital Transmission in the 21st Century: Conflating Modulation and Coding. *IEEE Communications Magazine*, pages 128–137, May 2002.

[9] F. J. MacWilliams, N. J. Sloane. *The Theory of Error-Correcting Codes*, volume 16. North-Holland, I edition, 1977.

[10] G.Caire, G.Taricco, E.Biglieri. Bit Interleaved Coded Modulation. *IEEE*, pages 1463–1466, 1997. Research supported by European Space Agency (ESA).

[11] Gottfried Ungerboek. Channel Coding with Multilevel/Phase Signal. *IEEE Transactions on Information Theory*, (1):55–67, January 1981.

[12] John G. Proakis. *Digital Communications*. McGraw-Hill, III edition, 1995.

[13] Joohee Kim, Russell M. Mersereau, Yucel Altunbasak. Error-Resilient Image and Video Transmission Over the Internet Using Unequal Error Protection. *IEEE TRANSACTIONS ON IMAGE PROCESSING*, 12(2):121–131, February 2003.

[14] Michael Gallant, Faouzi Kossentini. Rate-Distortion Optimized Layered Coding with Unequal Error Protection for Robust Internet Video. *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, 11(3):357–372, March 2001.

[15] Todd K. Moon. *Error Correction Coding — Mathematical Methods and Algorithms*. John Wiley and Sons, Inc., 1th edition, March 2005.

[16] P.Robertson and T.Worz. Bandwidth-efficient turbo trellis-coded modulation using punctured component codes. *IEEE J. Select. Areas Commun.*, 16(2):206–218, February 1998.

[17] Robert Gallagher. *Low-Density Parity-Check Codes*. Revisited Version, Massachussets Institute of Technology, Cambridge, Mass., July 1963.

[18] Roberto Garello, Franco Chiaraluce, Paola Pierleoni, Marco Scaloni, Sergio Benedetto. On Error Floor and Free Distance of Turbo Codes.

[19] S. Benedetto , R. Garello , G. Montorsi , C. Berrou , C. Douillard , D. Giancristofaro , A. Ginesi , L. Giugno , M. Luise . MHOMS: High Speed ACM Modem for Satellite Application. *IEEE Wireless Communications*, 2005.

[20] S.Fagioli, A.Capuzi, R.Leonardi, G.Angino. COSMO-SKYMED: The Italian Contribution to The French-Italian Inter-Governmental Agreement on Earth Observation. Customer Application Note, 2006.

[21] Stephen B. Wicker. *Error Control System for Digital Communication and Storage*. Prentice-Hall, 1995.

[22] Tom Richardson. The Renaissance of Gallager's Low-Density Parity-Check Codes. *IEEE Communications Magazine*, pages 126–131, August 2003.