Universitat Autònoma de Barcelona

Prototipatge Ràpid de la Capa Física d'OFDM: cas HIPERLAN/2

Memòria presentada per
Moisès Serra i Serra
per optar al grau de
Doctor en Informàtica

Directors:

Jordi Carrabina i Bordoll

Pere Martí i Puig

Bellaterra, Març de 2005

Abstract

This thesis presents the rapid prototyping design and methodology of a communication subsystem, specifically the digital part of the physical layer of wireless HiperLAN/2 protocol with the following contributions:

- Formulation and Validation of a design methodology that directly rely the system level specification and silicon platform validation with stress in heterogeneous multilevel verification
- Exploration and implementation of new hardware architectures to improve some complex algorithms in the HiperLAN/2 chain, obtaining synthesizable models for the digital part of the physical layer of both HiperLAN/2 transmitter and receiver validated in the hardware architecture.
- Prototyping the digital part of the physical layer of the HiperLAN/2 transmitter with a 12Mbits/s data transmission rate in a silicon platform, specifically sselected for WLAN systems. The aim of this prototype is to show the high productivity of the design methodology that ends into real measurable platforms..

Basically this thesis is structured into four parts as follows: fundamental concepts, the rapid prototyping methodology of a WLAN system, the physical layer design and prototyping of the transmitter.

Fundamental concepts introduce OFDM modulation, outlining the signal model as well as its advantages and disadvantages. Next, the standard wireless HiperLAN/2 that uses OFDM modulation is presented: working environments, network topology, physical layer, model's complexity and burst types of the HiperLAN/2 standard.

Rapid prototyping methodology for a WLAN system describes the design methodology used to prototype the transmitter and receiver of the digital part of the physical layer of HiperLAN/2, starting from the requirements of the WLAN system that let to specific prototyping platforms. In order to detain current HW implementation capabilities we present the architecture of Virtex FPGAs as core hardware device. Next, we present the WLAN system design methodology with the design flow and synchronization methods used. Two design flows are presented, a more general flow and a specific one oriented to WLAN systems. These design flows detail powerful alternatives for heterogeneous multilevel verification, that are based on the fundamental modern concepts of functional abstraction, orthogonality and heterogeneity.

Physical layer design of transmitter and receiver show their structures and detail different structural components. Significance contributions have been proposed for IFFT/FFT, channel equalizer, synchronization and Interleaver/Deinterleaver and Viterbi. Results obtained from the synthesis and simulation show their performance compared to current alternative designs. The overall results obtained from the transmitter and receiver models show also its global behavior and performance.

Transmitter prototyping details adaptations made to the transmitter model in order to map it into a specific silicon platform. Out of this complete prototype we can extract results obtained from the spectrum analyzer and transmitter's power consumption that validate at the end the approach proposed.

Índex

Figures

Taules

Capítol 1. **Introducció**

El disseny de sistemes electrònics ha evolucionat de manera espectacular durant la darrera dècada, a causa de l'increment de complexitat dels components que es podien integrar en una solució. Aquesta complexitat obliga a la introducció de metodologies de treballs en equip amb una divisió dels sistemes a dissenyar en diferents parts seguint criteris d'especialització (hardware, software, arquitectura SoC/NoC, comunicacions, DSP,...). Aquesta metodologia comporta uns majors requeriments d'especificació i verificació. En termes d'especificació, han aparegut en els darrers temps, un important conjunt de llenguatges, models i mètodes que permeten disposar d'especificacions més o menys executables a nivell de sistema. Pel que fa a la verificació, que es considera com un 80% de l'esforç que es dedica al disseny, inclou cada vegada més l'aplicació de tècniques de prototipatge ràpid, real o virtual, que permeten executar concurrentment models heterogenis (models SW, models de simulació del HW, codi HW i SW) executant-se sobre plataformes complexes constituïdes per ordinadors de propòsit general o estacions de treball, juntament amb plataformes en les quals es pot mapar els recursos HW i SW. Amb tot això, s'aconsegueix obtenir un producte final més ràpidament.

Aquestes metodologies s'han especialitzat segons els tipus de sistemes que s'han d'implementar i cada vegada més utilitzen els dispositius lògics programables (FPGAs). L'evolució tecnològica d'aquests dispositius ha portat a la introducció de noves tecnologies que es troben a la meitat del camí entre les FPGAs i els ASICS, com per exemple el HardCopy de Altera i EasyPath de Xilinx, que cobreixen un espai nou de producció que abans no existia. Un altre aspecte important a tenir en compte és la transformació que ha experimentat l'ús de les FPGAs que han passat de ser dispositius utilitzats per a prototipus simples o per a lògica senzilla d'aplicació específica a ser utilitzats en producte final, ja que per a moltes aplicacions és possible tenir el producte abans al mercat (típicament per a quantitats inferiors a 100.000 peces) amb la possibilitat d'actualització del hardware en cas de necessitat.

Els sistemes WLAN (*Wireless Local-Area Network*), en particular, les capes físiques dels estàndards l'HiperLAN/2 i IEEE802.11a, són sistemes complexos d'altes prestacions amb diferents tasses de transmissió de dades durant un mateix flux. Aquests sistemes incorporen algorismes amb una complexitat computacional elevada (FFT, IFFT, Viterbi) i amb unes restriccions temporals exigents.

En aquesta tesi es presenta el prototipatge ràpid d'un subsistema de comunicació sense fils, en concret, la part digital de la capa física de l'HiperLAN/2 amb els següents objectius.

- Adaptar una metodologia de disseny a nivell de sistema que ens permeti l'exploració i implementació de noves arquitectures hardware d'alguns dels algorismes més complexos de la cadena HiperLAN/2.
- Obtenir uns models sintetitzables per a FPGAs de la part digital de la capa física d'un emissor i d'un receptor HiperLAN/2, amb la finalitat de validar la metodologia i les arquitectures hardware dissenyades.
- Prototipar la part digital de la capa física de l'emissor HiperLAN/2 amb una tassa de transmissió de dades de 12Mbits/s en una plataforma específica per a sistemes DSP (*Digital Signal Processing*) o WLAN, per tal de validar el camí de baixar a placa des de Matlab establert per la metodologia.

1.1. Estructura de la tesi

Aquesta tesi està estructurada en sis capítols, el primer dels quals presenta una introducció a la tesi i l'organització d'aquesta memòria.

Al capítol 2, titulat conceptes fonamentals, s'introdueix els conceptes fonamentals de la modulació OFDM des d'una perspectiva orientada als sistemes de comunicacions sense fils que treballen en mode a paquets. Es presenta el model del senyal OFDM, la propietat d'ortogonalitat i els seus principals avantatges i inconvenients. A continuació, es presenta l'HiperLAN/2 que és un estàndard per a sistemes de comunicació sense fils orientat a paquets, el qual utilitza la modulació OFDM. En concret, s'explica la seva topologia, els entorns de treball, els tipus de ràfegues, les diferents capes de l'estàndard i en particular la capa física. S'acaba el capítol amb un petit sumari.

Al capítol 3, titulat metodologia de prototipatge ràpid d'un sistema WLAN, es fa una introducció al prototipatge ràpid, primerament justificant la necessitat del seu ús dins la metodologia de disseny i verificació, i detallant posteriorment les capacitats i propietats d'aquesta metodologia. A continuació, es presenta l'estat de l'art dels entorns i plataformes per modelar sistemes WLAN, així com una breu introducció als dispositius lògics programables, en concret, es comenta l'evolució de la família de FPGAs Virtex i es presenta l'arquitectura de la Virtex II. Llavors, es presenta la metodologia específica aplicada pel prototipatge d'un sistema WLAN. Dins la metodologia es fa especial esment del flux de disseny a nivell de sistema i la verificació heterogènia multinivell. Finalment s'acaba amb un sumari sobre el capítol.

Al capítol 4, titulat disseny de la capa física, s'exposa les característiques de l'emissor i receptor dissenyat. A continuació, s'expliquen els diferents blocs que formen l'emissor i el receptor. S'ha estructurat la presentació de cada bloc en diferents parts per fer més entenedora la seva explicació. Bàsicament s'explica el funcionament del bloc i si cal la seva justificació teòrica, es presenta el disseny del bloc amb les tècniques de disseny aplicades, es mostren els resultats obtinguts de síntesi i de simulació i s'exposen possibles alternatives de disseny. Per acabar es presenten els resultats i conclusions globals de l'emissor i receptor.

Al capítol 5, titulat prototipatge de l'emissor, es fa una introducció a la plataforma utilitzada en el prototipatge, es mostra el model d'emissor implementat i s'exposen els blocs necessaris afegits per tal d'implementar-lo a la plataforma de prototipatge. Després, es presenten les tècniques de rellotge utilitzades per tal de poder implementar l'emissor. Finalment, es mostren els resultats obtinguts d'ocupació, de potència i d'espectres d'OFDM.

Al capítol 6, titulat conclusions, es presenten les compilacions dels resultats sobre el model de l'emissor i receptor, les arquitectures explorades, la metodologia utilitzada i les línies de treball futur.

Capítol 2. **Conceptes fonamentals**

L'organització d'aquest capítol s'estructura en dos apartats que fan referència a la tècnica de modulació utilitzada i a un dels estàndards aplicats en les comunicacions sense fils en transmissions orientades a paquets.

A l'apartat 2.1 s'explica la tècnica de modulació OFDM (*Orthogonal Frequency Division Multiplexing*). Es comença exposant el model de senyal per a les modulacions OFDM amb les expressions matemàtiques que defineixen un símbol OFDM. Llavors, es fa incidència a tres aspectes importants d'aquestes modulacions: la propietat d'ortogonalitat que mantenen les subportadores, la generació d'un símbol OFDM digitalment a partir de l'equivalent passabaix utilitzant algorismes ràpids tipus FFT (*Fast Fourier Transform*) i es presenten els principals avantatges i desavantatges de l'ús d'OFDM.

A l'apartat 2.2 es comença fent una breu presentació a les bandes de freqüències per a sistemes sense fil de banda ampla i una breu introducció a les principals característiques dels diferents estàndards WLAN. Després es mostra l'estàndard HiperLAN/2 (*HIgh PErformance Radio Local Area Network type 2*) començant pel tipus d'entorn de treball pel qual ha estat pensat. A continuació es presenta la seva topologia, i també, es fa una breu introducció a les diferents capes utilitzades per HiperLAN/2 donant més èmfasis a la capa física. De la capa física es presenten els principals paràmetres segons les especificacions de l'estàndard HiperLAN/2 i els diferents blocs que formen l'emissor i receptor. Finalment, es presenten els cinc tipus de ràfegues que s'utilitzen.

2.1. Modulació OFDM

La tècnica de modulació OFDM és un cas particular de les modulacions multiportadora MCM (*Multi-Carrier Modulation*) i pot ser vista com una tècnica de modulació o com una tècnica de multiplexació [1][2][3].

2.1.1. Model de senyal

Un senyal OFDM, consisteix en una suma de subportadores que són modulades en fase PSK (*Phase Shift Keying*) o en amplitud QAM (*Quadrature Amplitude Modulation*). El seu principi de funcionament es basa en la conversió sèrie-paral·lel d'un flux de dades complexes d'alta velocitat c_i a un conjunt de N_s fluxos de dades en paral·lel de velocitats N_s vegades inferiors. Les dades c_i són el resultat de mapar un conjunt de dades binàries sobre un conjunt de possibles valors (PSK o QAM) en la constel·lació del senyal corresponent a cada flux. A continuació cadascun dels fluxos de dades es modula simultàniament sobre una subportadora diferent. Llavors, el temps del símbol OFDM resultant T s'incrementa en un factor N_s respecte al temps de símbol que es necessitaria per enviar el flux de dades sèrie original sobre una portadora única. Si anomenem f_c a la freqüència portadora, un únic símbol OFDM que comenci a l'instant $t = t_s$ es pot escriure com:

$$x(t) = \begin{cases} \mathrm{Re}\left[\sum_{i=-\frac{N_s}{2}}^{\frac{N_s}{2}-1} c_{\left(i+\frac{N_s}{2}\right)} e^{j2\pi\left(f_c-\frac{i+0.5}{T}\right)(t-t_s)} \right], \text{per } t_s \le t \le t_s + T \\ 0, \text{per } t < t_s \text{ i per } t > t_s + T \end{cases} \quad (1)$$

Si es representa *x(t)* a partir del seu equivalent passabaix *s(t)* s'obté una notació més compacta:

$$s(t) = \begin{cases} \sum_{i=-\frac{N_s}{2}}^{\frac{N_s}{2}-1} c_{\left(i+\frac{N_s}{2}\right)} e^{j2\pi\frac{i}{T}(t-t_s)} & , \text{per } t_s \le t \le t_s + T \\ 0, \text{per } t < t_s \text{ i per } t > t_s + T \end{cases} \quad (2)$$

El resultat és un senyal complex on la seva part real es coneix amb el nom de component en fase i la seva part imaginària amb el nom de component en quadratura. Per recuperar el senyal original *x(t)* a partir de l'equivalent passabaix *s(t)* cal multiplicar el component en fase i el component en quadratura pel cosinus i el sinus de la freqüència portadora respectivament. Normalment, a la literatura se sol escriure un únic símbol OFDM referit a l'origen (t_s = *0*). Llavors, la freqüència portadora s'escull de manera que l'equivalent passabaix *s(t)* que s'obté de l'equació (2) segueixi un indexat per a les portadores que vagi de *0* a N_s-*1*, segons l'expressió:

$$s(t) = \begin{cases} \sum_{i=0}^{N_s-1} C_i e^{j2\pi\frac{i}{T}(t)} & , \text{per } 0 \le t \le T \\ 0, \text{per } t < 0 \text{ i per } t > T \end{cases} \quad (3)$$

Per fer el sistema més robust a la propagació multicamí s'afegeix una extensió cíclica de longitud superior a la resposta impulsional. Aquesta tècnica evita totalment la interferència intersimbòlica ISI (*Inter Symbol Interference*) i millora considerablement la seva adaptació als canals dispersius en temps, però retalla lleugerament l'eficiència espectral de la modulació OFDM.

2.1.2. Ortogonalitat entre subportadores

Un senyal OFDM compleix una relació matemàtica precisa entre les freqüències de les subportadores. Des de una visió temporal de la propietat, a l'interval *T* totes les freqüències tenen exactament un número sencer de cicles. També, el nombre de cicles entre subportadores adjacents difereix exactament en una unitat. La següent figura mostra la representació de les primeres quatre subportadores en el domini temporal:

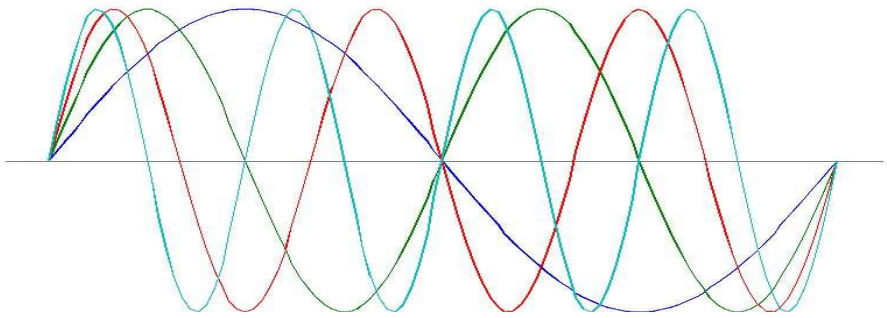


Figura 2.1 Ortogonalitat de les 4 primeres subportadores en el domini temporal

Una manera equivalent d'apreciar la propietat d'ortogonalitat és a partir de la representació freqüencial del senyal (Figura 2.2). L'espectre del conjunt és el resultat de convolucionar cadascuna de les deltes localitzades a les freqüències de les subportadores amb l'espectre d'un pols rectangular de durada *T* segons. L'amplitud de l'espectre del pols segueix una funció *sinc(π*f*T)*, amb zeros en totes les freqüències múltiples senceres de *1/T*. Llavors, quan una subportadora pren el valor màxim en el

domini freqüencial, en el mateix punt, la resta de subportadores s'anul·len, evitant així qualsevol interferència espectral entre elles.

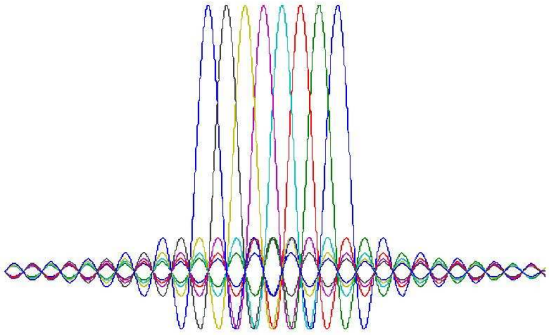


Figura 2.2 Espectres individuals de 8 subportadores OFDM

2.1.3. Generació d'un símbol OFDM utilitzant IFFT

Weinstein i Ebert van aplicar per primera vegada al 1971 la transformada discreta de Fourier (DFT) en el procés de modulació i desmodulació d'OFDM [8].

Si substituïm en l'expressió (3) la variable analògica *t* pels instants de mostratge nT/N_s per $n = 0,1,2,...N_s-1$, s'obté l'equivalent passabaix discret *s(n)*:

$$s(n) = \sum_{i=0}^{N_s-1} c_i e^{j2\pi\frac{in}{N_s}}, \text{per n} = 0,1,2,...\text{N}_\text{s} - 1 \quad (4)$$

Segons aquest resultat, es pot determinar que, llevat un terme constant, la transformació que s'està aplicant a les dades c_i és la Transformada Discreta Inversa de Fourier (IDFT). Aquesta observació va proporcionar un mètode eficient per calcular digitalment l'equivalent passabaix del senyal. Per recuperar les dades c_i del conjunt de mostres *s(n)*, en el procés de desmodulació, només cal realitzar l'operació inversa, en aquest cas la Transformada Discreta de Fourier (DFT). El receptor pot recuperar la informació sense la necessitat de filtrat i s'elimina la necessitat de disposar d'un banc d'oscil·ladors per a la generació de les subportadores.

Les transformades DFT i IDFT es realitzen de manera eficient a partir de la utilització dels corresponents algorismes ràpids FFT/IFFT, els quals redueixen el número de multiplicacions complexes necessàries que serien de l'ordre de N^2 a un valor orientatiu de $(N/2)\ log_2\ (N)$ utilitzant l'algorisme de radix-2 [2], gràcies a l'aprofitament de la redundància trobada en les operacions.

2.1.4. Avantatges i desavantatges de l'ús d'OFDM

L'esquema de modulació OFDM presenta molts avantatges respecte les tècniques que utilitzen una única portadora. A continuació se n'enumeren un conjunt.

- OFDM és una estratègia eficient per a combatre els efectes de la dispersió temporal del canal originada per la propagació multicamí.
- Es pot eliminar completament la ISI i la ICI (*Inter Carrier Interference*) amb les extensions cícliques adequadament dimensionades.
- És una tècnica robusta a les interferències de banda estreta perquè aquestes només afecten a una petita porció de les subportadores.

- Amb un correcte dimensionat de les extensions cícliques, OFDM combina de forma coherent els diferents feixos associats a la propagació multicamí, fet que es tradueix en un guany de diversitat en el receptor.
- La sincronització temporal és flexible i es pot produir dins un marge de valors possibles dins de les extensions cícliques sense degradar les prestacions del sistema.

Els punts febles d'aquesta tècnica respecte als sistemes basats en una única portadora són bàsicament tres:

- Efectes dels errors de sincronisme freqüencial. La gran sensibilitat als errors de freqüència i al soroll de fase sembla ser la limitació més important. Aquests fenòmens trenquen la ortogonalitat entre les subportadores i originen interferències ICI. Cal habilitar una tècnica d'estimació i correcció de les desviacions de freqüència prou acurada per mantenir el sistema en marges de bon funcionament.
- Efecte dels pics de potència. El senyal OFDM és d'envolupant no constant i en determinades ocasions poden aparèixer pics intensos que porten els amplificadors de radiofreqüència a treballar en zones poc lineals.
- Radiació fora de bandes. Les característiques espectrals del senyal poden no decréixer suficientment ràpid fora de la banda de radiació i originar interferències a altres sistemes, cosa que es pot combatre enfinestrant el senyal.

2.2. HiperLAN/2

És important abans de començar amb l'estàndard HiperLAN/2 fer una breu exposició de les bandes de freqüència lliures utilitzades per als sistemes de comunicació sense fil de banda ampla i dels principals estàndards WLAN.

Hi ha tres fòrums principals que controlen l'estandardització i les bandes de freqüència dels sistemes de comunicació sense fils de banda ampla, anomenats IEEE 802.11, ETSI BRAN (*European Telecommunications Standards Institute Broadband Radio Access Networks*) i MMAC (*Multimedia Mobile Access Communications*). La Figura 2.3 mostra les bandes de freqüència (superiors als GHz) utilitzades a Europa, Estats Units i Japó.

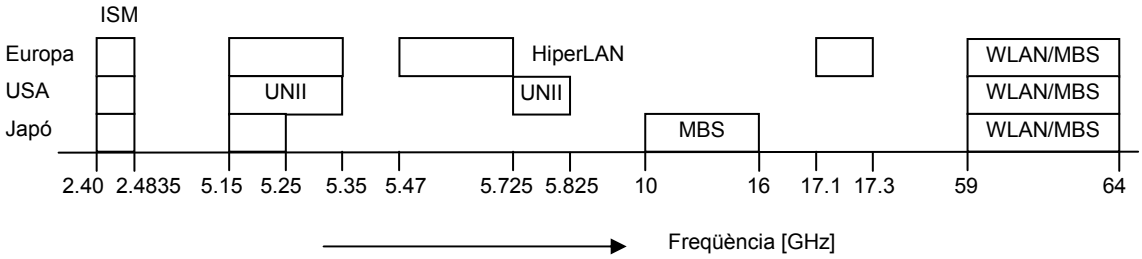


Figura 2.3 Bandes de freqüència per a comunicacions sense fils de banda ampla

La banda de 2.4GHz és una banda per aplicacions Industrials, científiques i mèdiques (ISM). La banda de 5GHz és designada específicament per a sistemes de comunicació sense fil de banda ampla. Cal destacar que en aquesta banda Japó només disposa de 100MHz, mentre que Estats Units i Europa disposen de 300 i 455MHz respectivament. A la banda de 17GHz Europa disposa d'un espectre extra per a HiperLAN mentre que Japó disposa d'una banda de 10 a 16GHz per a sistemes mòbils de banda ampla (MBS).

A continuació es fa una breu presentació de les principals característiques dels principals estàndards WLAN. Per començar la Taula 2.1 mostra una comparativa entre els diferents estàndards WLAN.

Estàndard	802.11a	802.11b	802.11g	802.11h	HiperLAN/2	*Bluetooth*
Organisme	IEEE	IEEE	IEEE	IEEE	ETSI	Bluetooth/SIG
Finalització	2002	1999	2003	2003	2003	2002
Banda de freqüència	5GHz	2,4GHz	2,4GHz	5GHz	5GHz	2,4GHz
Velocitat màxima	54Mbits/s	11Mbits/s	54Mbits/s	54Mbits/s	54Mbits/s	0.721Mbits/s
Tipus de modulació	OFDM	DSSS	DSSS/OFDM	OFDM	OFDM	FHSS

Taula 2.1 Comparativa entre diferents estàndards WLAN

IEEE 802.11

Aquest va ser el primer estàndard definit per IEEE i va ser publicat l'any 1977. Funciona sobre la banda de 2,4 GHz (de 2.400 MHz a 2.483,5 MHz) i utilitza dos tipus de modulació: DSSS (*Direct Sequence Spread Spectrum*) i FHSS (*Frequency Hopped Spread Spectrum*). La velocitat de transmissió pot arribar a 2Mbits/s.

IEEE 802.11b

Aquest és l'evolució natural de l'anterior estàndard. Bàsicament, es diferencien en l'ús exclusiu de la modulació DSSS amb el sistema de codificació CCK (*Complementary Code Keying*). Les velocitats que permet assolir aquest estàndard són 1, 2, 5.5 i 11Mbits/s.

IEEE 802.11a

També, aquest estàndard és una evolució del 802.11 i va ser aprovat l'any 1999 i també es coneix amb el nom de "*Wi-Fi5*". Aquest estàndard treballa a la banda dels 5GHz (5.150MHz a 5.350 MHz i 5.470 MHz a 5.725 MHz) i utilitza la tècnica de modulació OFDM. Les velocitats de transmissor van dels 6 als 54Mbits/s.

IEEE 802.11g

Aquest estàndard que es basa amb la norma 802.11b i va ser aprovat a l'any 2003. Treballa a la banda dels 2.4GHz i es capaç d'utilitzar els dos mètodes de modulació (DSSS i OFDM). La velocitat de transmissió pot arribar fins els 54Mbits/s.

IEEE 802.11h

És una evolució del IEEE 802.11a que permet l'assignació dinàmica de canals i control automàtic de potència per minimitzar els efectes de possibles interferències.

BLUETOOTH

El *Bluetooth* és un enllaç de ràdio de poc abast (10m/100m) que s'associa a les xarxes d'àrea personal sense fils o WPAN (*Wireless Personal Area Network*). El *Bluetooth* treballa a la banda de 2.4GHz (2,402 a 2,480 GHz) i utilitza la modulació FHSS. La velocitat de transmissió de dades pot arribar fins els 0.721Mbits/s

Home RF

Aquesta tecnologia ha estat definida per un grup de companyies sota el nom de HRFWG (*Home RF Working Group*). El Home RF utilitza la mateixa banda de 2.4GHz, però disposa d'un mètode de salt de freqüència SWAP(*Shared Wireless Access Protocol*) per no interferir amb les connexions *Bluetooth*. La velocitat de transmissió de dades pot arribar fins els 2Mbits/s.

HiperLAN/2

És un estàndard pensat per a xarxes d'abast local sense fils (WLAN) orientades a paquets proposat per l'ETSI BRAN, el qual utilitza la modulació OFDM. Proporciona altes velocitats de transmissió, de fins a 54 Mbit/s, a terminals mòbils en distàncies curtes, de fins a 150 metres, en ambients tant d'interior com d'exterior. Disposa de 19 canals situats a la banda de freqüència de 5GHz (5.150 a 5.350GHz i 5.470 a 5.725GHz).

2.2.1. Entorns de treball HiperLAN/2

HiperLAN/2 ha estat desenvolupat pensant en dos entorns en concret, l'entorn domèstic i l'entorn de "negocis" (àrees grans com aeroports, universitats, hospitals, ...), els quals determinen dos tipus de xarxa: *Domestic Premises Network* (DPN) i *Business Premises Network* (BPN).

La BRAN (*Broadband Radio Acces Networks*) defineix cinc models de canal (definits a la Taula 2.2) per ser utilitzats en les simulacions dels sistemes WLAN en funció dels entorns físics equivalents [42]. També és interessant el llibre *Wireless OFDM Systems* que dedica tot un capítol a explicar els models de propagació de radio per a entorns tancats (*indoor*) [43].

Canal	RMS delay Spread	Característiques	Entorn de treball
A	50ns	Rayleigh	Oficina, NLOS (no line of sight)
B	100ns	Rayleigh	Espai obert/oficina, NLOS
C	150ns	Rayleigh	Espai obert gran, NLOS
D	140ns	Rice	Com el canal C amb LOS
E	250ns	Rayleigh	Espai obert gran, NLOS

Taula 2.2 Models de canal Hiperlan/2

2.2.2. Topologia de la xarxa HiperLAN/2

L'HiperLAN/2 és un sistema cel·lular que està format per dos tipus de dispositius: el terminal mòbil (MT) i el punt d'accés (AP) en entorns de negocis o el controlador central (CC) en entorns domèstics. Disposa de dos modes d'operació. El mode centralitzat, la comunicació dels terminals mòbils amb la xarxa o altres mòbils passa a través del punt d'accés El mode directe, existeixen enllaços directes entre terminals mòbils. En tots els casos, el punt d'accés assigna els recursos de radio i controla les comunicacions a les cèl·lules.

En entorns de negocis, la majoria del tràfic passa a través del punt d'accés, però també hi ha la possibilitat de transferència de dades entre terminals "*Direct Mode*". En aquest cas, el punt d'accés es limita al control d'intercanvi de trames PDU (*Protocol Data Unit*) entre terminals. En el cas d'entorns domèstics, la majoria del trànsit passa entre terminals [7].

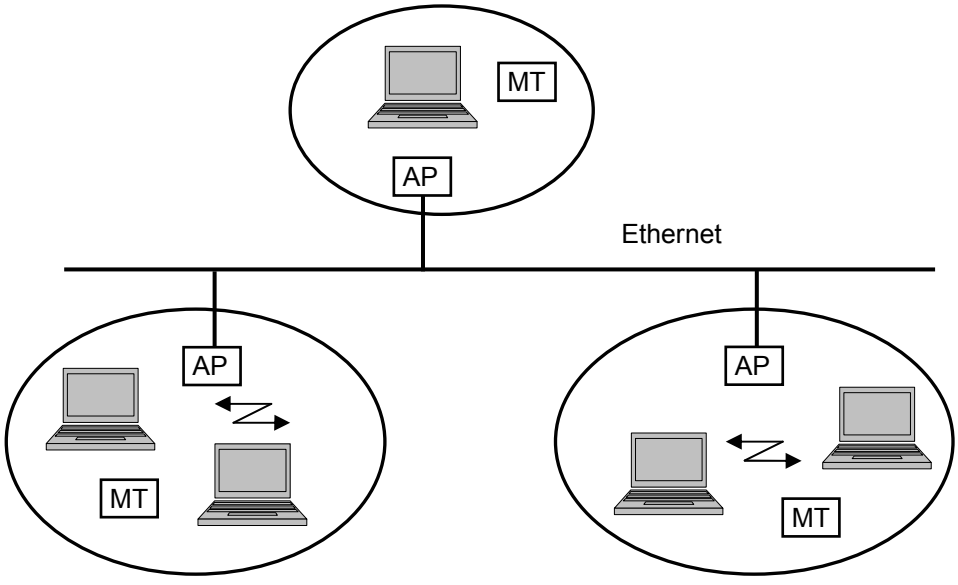


Figura 2.4 Topologia bàsica d'una xarxa HiperLAN/2

2.2.3. Introducció a la capa física

HiperLAN/2 pot ser utilitzat en una varietat de serveis i protocols. Això és possible mercès a la seva arquitectura flexible (Figura 2.5) que defineix una capa de convergència entre aquests protocols de les capes superiors i la capa de control d'enllaç de dades (DLC) de l'HiperLAN/2 [5]. La capa de convergència (CL) suporta varis protocols: IP (*Internet Protocol*), ATM (*Asynchronous Transfer Mode*), IEEE 1394. La capa CL té dues principals funcions. La primera, és passar els serveis demanats per les capes superiors als serveis oferts per la capa DLC. La segona, és segmentar els diferents formats de dades que arriben de les capes superiors en unitats de dades de longitud fixa, perquè puguin ser tractades per la capa DLC i viceversa. Els blocs de dades de longitud variable que arriben de les capes superiors, coneguts amb el nom de unitats de servei de dades (SDU), són segmentats en unitats de 384 bits. Aquestes unitats, conjuntament amb unes capçaleres d'informació, són empaquetades amb protocols d'unitats de dades anomenats PDU. Llavors, aquests nous paquets són enviats a la capa DLC (Figura 2.6).

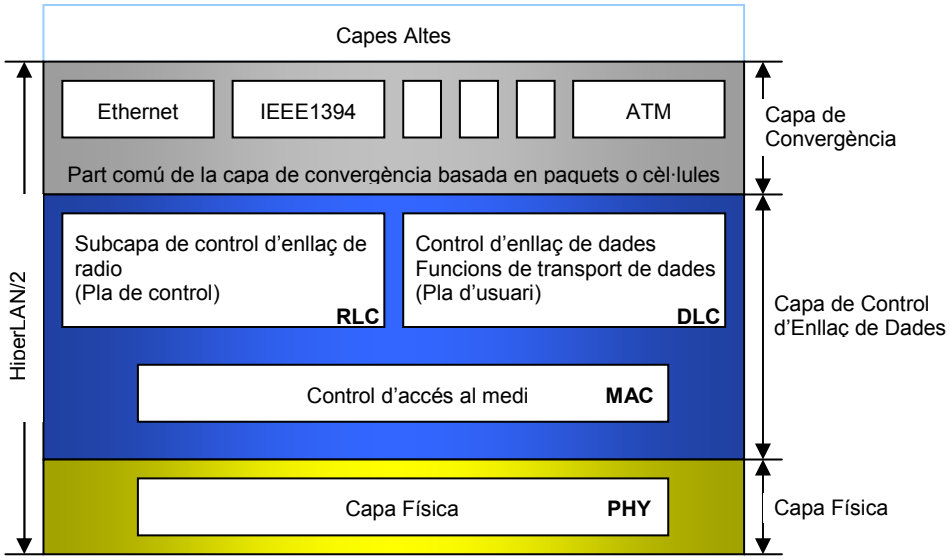


Figura 2.5 Capes de l'HiperLAN/2

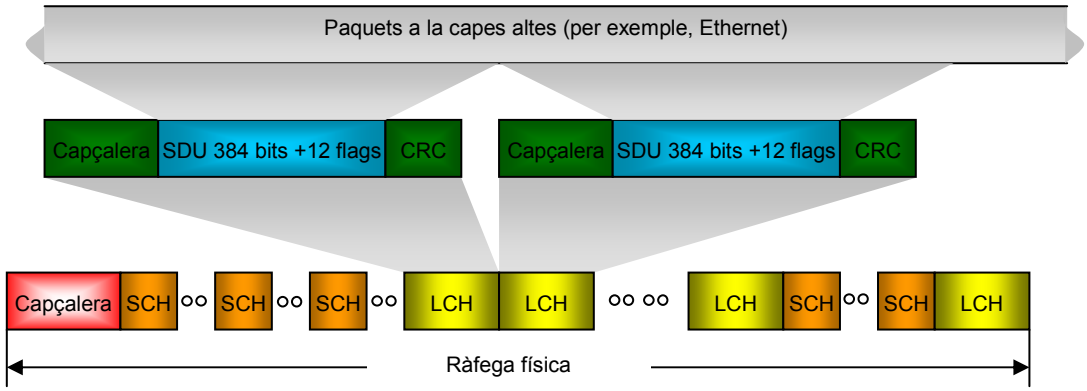


Figura 2.6 Segmentació i empaquetament

La capa DLC constitueix l'enllaç lògic entre els dispositius HiperLAN/2. Una de les funcions d'aquesta capa és el control a l'accés al medi (MAC). L'accés al medi de HiperLAN/2 es basa en el protocol TDMA/TDD (*Time Division Multiple Acces/Time Division Duplexing*). Aquest protocol es basa en una trama MAC bàsica amb una durada fixa de 2ms dividida en diversos *slots* temporals (Figura 2.7).

La capa MAC tracta tota la informació en tres formats: canals lògics, canals de transport i seqüència de PDUs.

Els canals lògics s'utilitzen per a la comunicació del nivell físic amb els nivells superiors. Aquests són definits per a diferents tipus de serveis de transferència de dades oferts per l'entitat MAC, en concret HiperLAN/2 disposa de 10 canals lògics especificats en l'estàndard de la capa DLC [5]. Cada tipus de canal lògic es defineix a través del tipus d'informació que porta i de la interpretació dels valors en els corresponents missatges.

A continuació, els canals lògics són mapats sobre els canals de transport. Els canals de transport defineixen el format de les dades a transmetre a través de la interfície d'aire i amb quines característiques es transmeten, mentre el contingut del missatges i la seva interpretació són subjectes als canals lògics. Així, els canals de transport es poden considerar com un tipus d'enllaç d'informació lliurada des de les capes altes a la capa física. El canals de transport que poden formar una trama MAC són (Figura 2.7):

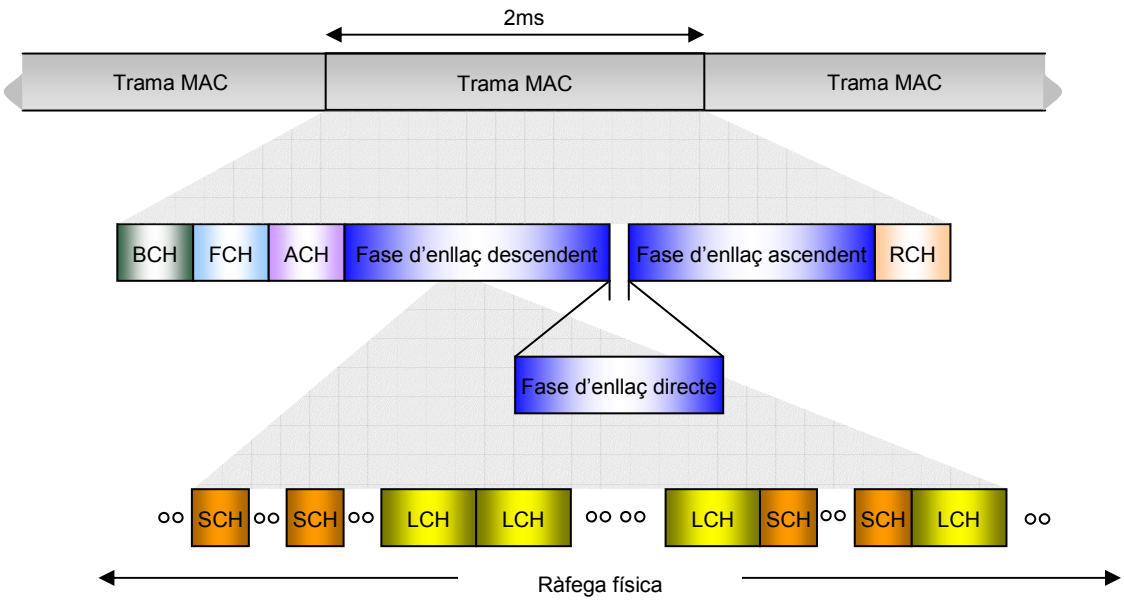


Figura 2.7 Trama bàsica de HiperLAN/2

El canal BCH (*Broadcast Channel*). S'envia des del punt d'accés a tots els terminals mòbils al principi de cada trama MAC i conté informació de tota la cèl·lula de radio (potència de transmissió i recepció del AP, punters i informació sobre els altres canals de transport de la trama).

El canal FCH (*Frame Channel*). S'envia des del punt d'accés a tots els terminals mòbils. La informació que porta descriu l'estructura de la interfície amb l'aire de la trama MAC.

El canal ACH (*Access Feedback Channel*). S'envia des del punt d'accés a tots els terminals mòbils. S'utilitza per respondre als terminals que havien fet una demanda durant l'accés aleatori RCH anterior.

El canal LCH (*Long Channel*). És bidireccional i s'utilitza per enviar dades de l'usuari en les fases d'enllaç descendent i ascendent.

El canal SCH (*Short Channel*). És bidireccional i s'utilitza per enviar informació de control en les fases d'enllaç descendent i ascendent.

El canal RCH (*Random Access Channel*). S'envia des del terminal mòbil al punt d'accés per transportar les demandes de recursos en les fases d'enllaç ascendent i directe.

El número de *slots* per a cadascun dels canals de transport en la trama és variable en funció dels requeriments de cada canal. Només el canal BCH disposa d'un interval fix de temps d'ocupació en la trama.

Llavors, els canals de transport són concatenats per construir els trens PDUs (Figura 2.7) . La seqüència de PDUs s'utilitza en intercanvis de dades amb la capa física. Les unitats de dades que s'han de transmetre a través de la capa física de HiperLAN/2 són ràfegues de longitud variable. En totes les fases, el format de la transmissió a la capa física és una ràfega que es composa d'una capçalera i un camp de dades. Finalment, la capa física mapa la seqüència de PDUs lliurats per la capa MAC sobre les ràfegues físiques.

2.2.4. Capa física

L'estructura de l'emissor del HiperLAN/2 es basa en les recomanacions de l'estàndard [4]. No obstant, l'estructura i les especificacions del receptor no hi són incloses.

Els principals paràmetres de la capa física segons les especificacions de l'estàndard HiperLAN/2 es poden veure a la Taula 2.3.

Velocitat de transmissió	6,9,12,18,27,36 i 54 MBits/s
Modulació	BPSK, QPSK, 16-QAM, 64-QAM
Relació de codificació	1/2 ,9/16,3/4
Número de subportadores	52
Número de pilots	4
Durada del símbol OFDM	4 µs
Durada del prefix cíclic	800ns
Espai entre subportadores	312,5 KHz
Amplada de banda del senyal	16,66MHz
Freqüència de mostreig	20MHz

Taula 2.3 Principals paràmetres de la capa física HiperLAN/2

Les capes físiques de l'IEEE 802.11a [6] i HiperLAN/2 [4] són bastant iguales, a part de les capçaleres de les ràfegues i d'alguna velocitat de transmissió, tal com mostra la següent taula.

Mode	Vel. (Mbit/s)	N_{DBPS} (bit)	Modulació	Rel. Cod.	N_{BPSC} (bit)	N_{CBPS} (bit)
1	6	24	BPSK	½	1	48
2	9	36	BPSK	¾	1	48
3	12	48	QPSK	½	2	96
4	18	72	QPSK	¾	2	96
5 (només H/2)	27	108	16QAM	9/16	4	192
5 (només IEEE)	**24**	**96**	**16QAM**	**1/2**	**4**	**192**
6	36	144	16QAM	¾	4	192
7	54	216	64QAM	¾	6	288
8 (només IEEE)	**48**	**192**	**64QAM**	**2/3**	**6**	**288**

Taula 2.4 Modes de transmissió HiperLAN/2 i IEEE 802.11a

A la Taula 2.4 es mostren els diferents modes possibles de transmissió d'HiperLAN/2 i IEEE 802.11a. A la taula s'han diferenciat els modes que permet HiperLAN/2 i IEEE 802.11a. Cal destacar dels modes d'HiperLAN/2 que els sis primers són obligatoris i l'últim de 64-QAM és opcional. N_{BPSC} i N_{CBPS} són el número de bits codificats per a cada subportadora i el número de bits codificats per símbol OFDM

respectivament. Cal remarcar que segons el mode seleccionat la velocitat de transmissió de dades pot variar entre 6 i 54Mbits/s i el número de bits de dades per símbol OFDM (N_{DBPS}) entre 24 i 216 bits. Aquests valors s'aconsegueixen combinant els diferents tipus de modulació (BPSK, QPSK i QAM) amb les diferents relacions de codificació (1/2, 3/4 i 9/16).

La cadena de transmissió HiperLAN/2 es pot veure a la Figura 2.8. En el camí de l'emissor, les dades arriben de la capa DLC al bloc *Data Scrambler*. Aquestes dades (PDU train) són una seqüència de 0 i 1. El bloc *Data Scrambler* modifica el bits per evitar llargues seqüències de 0 o 1 reduint la tassa de PAP (*Peak-to-Average Power*), ja que valors alts de PAP incrementen la complexitat del sistema i encareixen l'etapa de conversió AD/DA i l'etapa de l'amplificador de potència que ha de treballar en zona lineal en un rang d'amplituds major.

Les dades arriben al bloc FEC on són codificades en una relació ½ per un codificador convolucional augmentant la redundància. Les altres relacions de codificació que es mostren en la Taula 2.4, s'aconsegueixen a través del bloc *punturing* a la sortida del codificador convolucional. Aquest bloc s'encarrega d'eliminar una sèrie de bits de la cadena de dades per aconseguir la relació de codificació desitjada.

Els bits codificats arriben al bloc *Interleaver*. Aquest bloc desordena els bits per evitar ràfegues d'errors intercanviant-los entre les subportadores que formen un símbol OFDM, ja que quan els esvaïments freqüencials són profunds algunes d'aquestes portadores poden resultar greument afectades causant errors que es succeeixen en ràfegues. Després, les dades passen al bloc *Mapper* on els bits són modulats en fase o en amplitud segons el mode que es vulgui treballar (Taula 2.4). Els bits són dividits en grups de 1 ,2 ,4 o 6 bits i convertits en símbols complexos BPSK, QPSK, 16QAM o 64QAM.

Per facilitar la recepció, el bloc *Pilot Insertion* afegeix quatre símbols pilot per a cada 48 símbols complexos. Llavors, aquests símbols són modulats sobre 64 subportadores mitjançant el bloc IFFT. Cal esmentar que solament 52 subportadores porten informació, les 12 restants no s'utilitzen. A continuació, per fer el sistema més robust a la propagació de camins, s'afegeix un prefix cíclic al principi del símbol que és una copia de les 16 últimes mostres d'ell mateix.

Finalment, abans de convertir la sortida digital a analògica, el bloc *Preambles* afegeix una capçalera per a cada ràfega. L'HiperLAN/2 defineix quatre tipus de capçaleres en funció de la fase que es trobi la transmissió. Aquestes capçaleres serveixen al receptor per fer la sincronització en temps, estimació d'offset de freqüència, control de guany i estimació de canal.

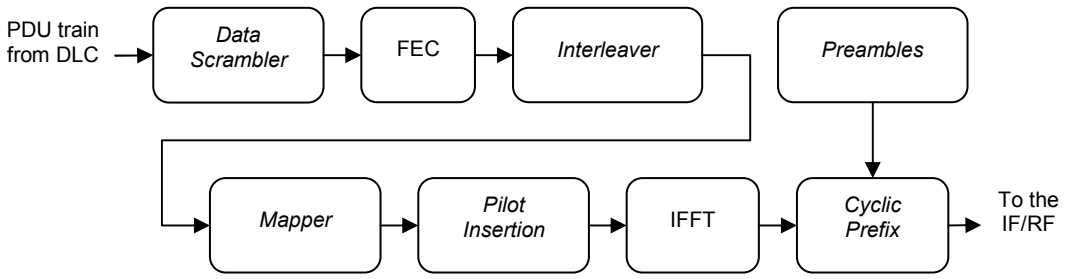


Figura 2.8 Cadena d'emissió HiperLAN/2

La Figura 2.9 mostra la cadena de recepció HiperLAN/2. Seguint la cadena de recepció, les dades arriben al bloc de sincronització. Aquest bloc realitza dues tasques. La primera, és trobar a on comença la trama i la segona, és estimar i corregir

l'*offset* de freqüència del senyal rebut. Per realitzar aquestes tasques s'utilitzen les capçaleres que envia a l'emissor.

Després de la sincronització, s'extrau el prefix cíclic per tal d'aconseguir les 64 mostres de cada símbol OFDM i es desmodula el senyal rebut mitjançant el bloc FFT.

Un cop desmodulat el senyal OFDM, s'estima el canal per tal de poder-lo equalitzar. L'estimador de canal compara el senyal rebut amb una seqüència coneguda, amb l'objectiu d'estimar la fase i l'atenuació d'amplitud del senyal rebut deguts al canal. Per realitzar l'estimació de canal, l'estàndard utilitza una part de les capçaleres enviades per l'emissor. A més, s'utilitzen els símbols pilots inserits en l'emissor per anar corregint la fase de les dades rebudes.

Finalment, les dades passen pels blocs *Demapper*, *Deinterleaver*, FEC i *Descrambler*, els quals realitzen la funció inversa de l'explicada anteriorment en l'emissor.

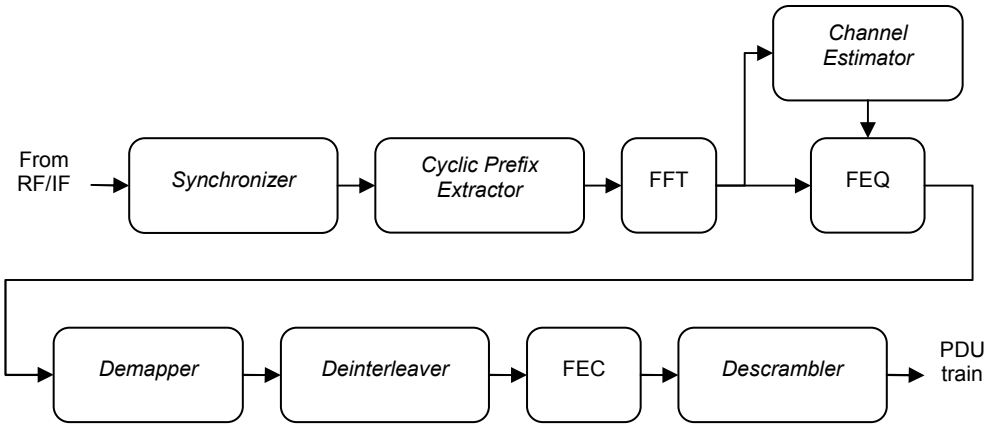


Figura 2.9 Cadena de recepció HiperLAN/2

2.2.5. Complexitat del model

Per fer una valoració òptima de la metodologia i tecnologia més adequada per prototipar o implementar un disseny, és important conèixer la seva complexitat. Una de les maneres de mesurar aquesta complexitat és fent una estimació del número d'operacions bàsiques necessàries per segon (MOPS). Llavors, per fer aquesta estimació de MOPS s'ha agafat l'estimació feta per en K. Masselos [9] d'un sistema HiperLAN/2. En aquesta estimació s'ha dividit el sistema per tasques o blocs que formen l'emissor i receptor HiperLAN/2. Aquesta tasques coincideixen amb els blocs presentats en l'apartat anterior de l'emissor i receptor menys els blocs de *puncturing*, *depuncturing*, codificador convolucional i descodificador de Viterbi que queden inclosos en el bloc FEC. Cal remarcar que les operacions bàsiques incloses en la estimació són operacions aritmètiques, lògiques i accessos de lectura/escriptura a la memòria. Per fer aquesta estimació més simple, s'assigna el mateix pes a totes aquestes operacions.

El resultat d'aquesta estimació es pot veure en les dues taules següents corresponents a l'emissor i receptor (Taula 2.5 i Taula 2.6). Aquestes taules mostren la complexitat del model pels set modes possibles de treball de l'HiperLAN/2. Cal destacar d'aquests resultats que la complexitat a l'emissor arriba fins a 4.164MOPS i al receptor fins a 13.781MOPS. Per tant, la suma totals de MOPS per tot el sistema s'aproxima als 18GOPS. També és interessant saber que els algorismes amb més complexitat són les IFFT/FFT i el descodificador de Viterbi, els quals poden tenir un cert grau elevat de paral·lelisme.

Aquests resultats d'estimació avalen la utilització de FPGA's en el prototipatge d'aquests sistemes, ja que aquests dispositius són ideals per arquitectures que treballin en paral·lel. Per exemple les últimes FPGAs de Xilinx poden arribar a superar els 20GOPS [17].

	Complexitat computacional / mode (MOPS)						
Tasca	**6 Mb/s**	**9 Mb/s**	**12 Mb/s**	**18 Mb/s**	**27 Mb/s**	**36 Mb/s**	**54 Mb/s**
Scrambler	108	162	216	324	486	648	972
Codificador convolucional	174	261	348	522	783	1044	1566
Puncturing 1	0.31	0.31	0.31	0.31	0.31	0.31	0.31
Puncturing 2	0	33	0	66	105	132	198
Interleaver	48	48	96	96	192	192	288
Mapat	30	45	36	54	54	72	90
Inserir Pilots	56	56	56	56	56	56	56
IFFT	922	922	922	922	922	922	922
Inserir prefix	72	72	72	72	72	72	72
Suma	**1410**	**1599**	**1746**	**2112**	**2670**	**3138**	**4164**

Taula 2.5 Complexitat computacional d'un emissor HiperLAN/2

	Complexitat computacional / mode (MOPS)						
Tasca	**6 Mb/s**	**9 Mb/s**	**12 Mb/s**	**18 Mb/s**	**27 Mb/s**	**36 Mb/s**	**54 Mb/s**
Descrambler	108	162	216	324	486	648	972
Viterbi	1170	1755	2340	3510	5265	7020	10530
Depuncturing 1	0.16	0.20	0.16	0.20	0.28	0.20	0.20
Depuncturing 2	0	50	0	99	118	198	297
Deinterleaver	48	48	96	96	192	192	288
Desmapat	48	48	240	240	288	288	336
Equalitzador	132	132	132	132	132	132	132
FFT	922	922	922	922	922	922	922
Extracció prefix	96	96	96	96	96	96	96
Sincronitzador	208	208	208	208	208	208	208
Suma	**2732**	**3421**	**4250**	**5627**	**7707**	**9704**	**13781**

Taula 2.6 Complexitat computacional d'un receptor HiperLAN/2

2.2.6. Tipus de ràfegues

Segons les especificacions de l'estàndard HiperLAN/2, hi ha diverses capçaleres que són inserides a l'emissor el principi de cada ràfega. Els cinc tipus de ràfegues són:

- Control i difusió
- Enllaç descendent
- Enllaç ascendent amb capçalera curta
- Enllaç ascendent amb capçalera llarga
- Enllaç directe (opcional)

Independentment del tipus de ràfegues, cada ràfega consisteix d'una capçalera (*rpreamble*) seguida de les unitats de dades (PDU) convertides a símbols OFDM (*rpayload*). Les diferents seccions de les capçaleres estan formades per símbols curts A, IA, B i/o IB, on IA i IB són les còpies invertides de A i B respectivament en el domini del temps. També, hi ha seccions amb símbols C que estan formats per 2 símbols OFDM de 64 mostres i un prefix cíclic CP de 32 mostres. A continuació només es mostren les diferents estructures que formen les capçaleres. Els passos per construir el símbols A,B i C s'especifiquen a l'estàndard HiperLAN/2 [4].

Ràfega de control i difusió

La ràfega de control i difusió conté informació de control de la trama i és transmesa des d'un punt d'accés a tots el terminals mòbils de la cèl·lula. La longitud de la capçalera és de 16μs i la seva estructura es pot veure a la Figura 2.10.

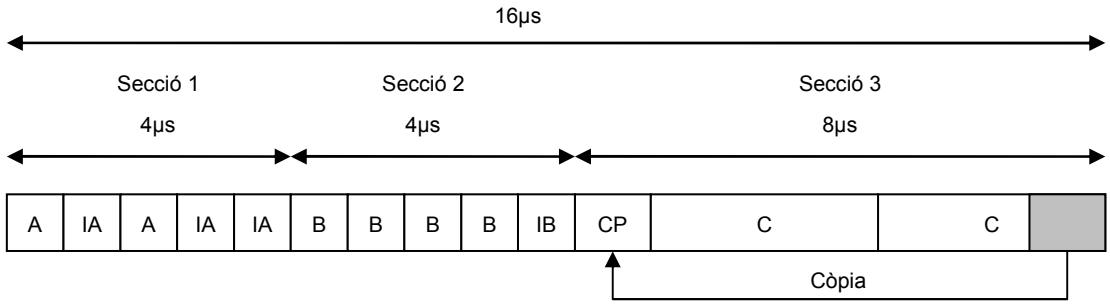


Figura 2.10 Capçalera per a una ràfega de Control i Difusió

Ràfega d'enllaç descendent

La ràfega d'enllaç descendent conté dades i informació de control que són transmeses des del punt d'accés a cadascun dels terminals mòbils individualment. La longitud de la capçalera és de 8μs i la seva estructura es pot veure a la Figura 2.11.

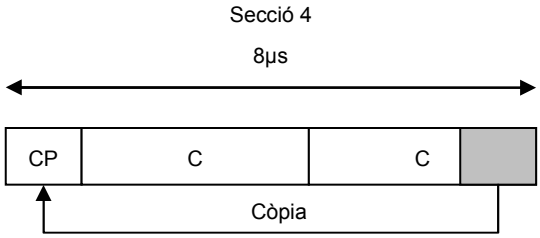


Figura 2.11 Capçalera per a una ràfega d'enllaç descendent

Ràfega d'enllaç ascendent amb capçalera curta

La ràfega d'enllaç ascendent amb capçalera curta conté dades i informació de control que són transmeses des de cadascun dels terminals mòbils al punt d'accés. La longitud de la capçalera és de 12μs i la seva estructura es pot veure a la Figura 2.12.

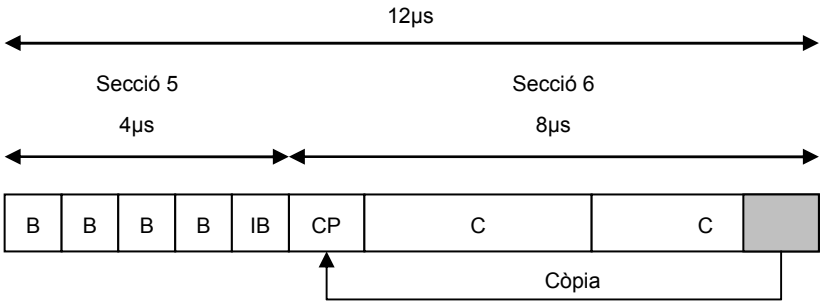


Figura 2.12 Capçalera curta per a una ràfega d'enllaç ascendent

Ràfega d'enllaç ascendent amb capçalera llarga

La ràfega d'enllaç ascendent amb capçalera llarga conté dades i informació de control que són transmeses des de cadascun dels terminals mòbils al punt d'accés. La longitud de la capçalera és de 16μs i la seva estructura es pot veure a la Figura 2.13.

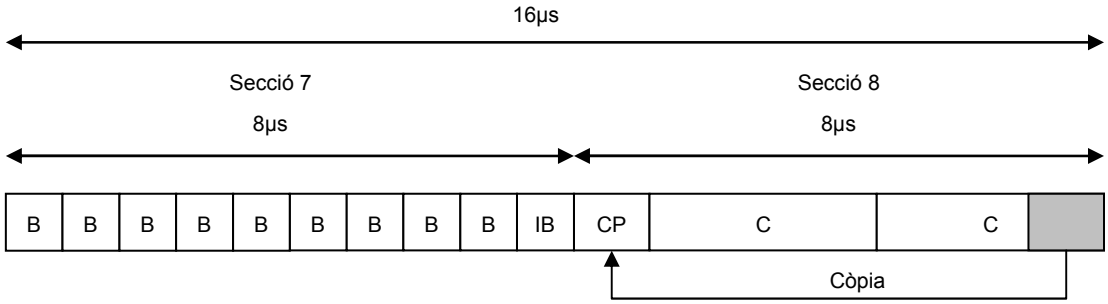


Figura 2.13 Capçalera llarga per a una ràfega d'enllaç ascendent

Ràfega d'enllaç directe

La ràfega d'enllaç directe s'específica com opcional i la tranferència entre dades és directe entre terminals mòbils. La longitud de la seva capçalera i la seva estructura són idèntiques que la ràfega d'enllaç ascendent amb capçalera llarga.

2.3. Sumari

En aquest capítol s'ha fet una introducció a conceptes fonamentals que són necessaris per als pròxims capítols. A més, alguns d'aquests conceptes faciliten al lector a entendre i a situar-se millor en la tesi. S'ha introduït la modulació OFDM utilitzada en l'estàndard HiperLAN/2. S'ha exposat els entorns de treball, topologia de la xarxa, la capa d'accés al medi i el tipus de ràfegues relacionats amb la capa física de HiperLAN/2. En concret de la capa física s'ha fet una breu descripció de la funcionalitat de cadascun dels blocs que formen l'emissor i receptor de l'HiperLAN/2. També s'han presentat les especificacions físiques, temporals i de complexitat de la capa física d'HiperLAN/2. Cal tenir en compte que aquestes especificacions ens serveixen de partida per seleccionar la metodologia de prototiptage ràpid més adequada i pel disseny de la capa física en els propers capítols.

Capítol 3. Metodologia de prototipatge ràpid d'un sistema WLAN

Aquest capítol s'ha estructurat en cinc apartats. En primer lloc, es fa una breu introducció al prototipatge ràpid justificant la seva utilització dins la metodologia de disseny de sistemes electrònics. Al segon apartat, es detallen els requeriments de la plataforma de prototipatge per a sistemes WLAN, que inclou els diferents aspectes que faciliten el disseny basat en plataformes: la metodologia, l'arquitectura i la tecnologia. Al tercer apartat, es fa una anàlisi de les opcions actuals d'entorns i plataformes de treball per modelar sistemes WLAN a nivell sistema. Al quart apartat, es fa una introducció a les FPGAs i una breu descripció de l'evolució de les arquitectures de les FPGAs Virtex de Xilinx. Al cinquè apartat, es presenta la metodologia proposada per desenvolupar un subsistema WLAN, passant pels mètodes de sincronització, fluxos de disseny, verificació, quantificació, abstracció funcional, ortogonalitat i heteregoneïtat. S'acaba amb un sumari del capítol.

3.1. Introducció al prototipatge ràpid

Prototipatge és definit per l'IEEE com: "Un tipus de desenvolupament que posa èmfasis en dissenyar prototipus ràpids en el procés de disseny, el qual permet una realimentació i anàlisi ràpida en suport d'aquest procés" [10].

Per detallar l'idea de prototipatge primer és necessari saber què és un prototipus. Un prototipus és un model d'un sistema que reflecteix amb exactitud un subconjunt determinat de les seves propietats. Hi ha prototips de tots tipus, essent els més populars els utilitzats per a mostrar propietats mecàniques (p.e. cotxes o avions), associades als disseny funcional o estètic dels productes. Pel que fa als prototips de sistemes electrònics, les propietats més usuals que es volen reflectir solen ser funcionals: resultats de computació o formats de presentació, o de prestacions: respostes temporals, estimació d'àrea de silici, etc. L'objectiu del prototipatge és aconseguir una funcionalitat del sistema la més propera possible a la desitjada, amb el propòsit de validar algorismes i arquitectures i determinar les característiques i els costos del sistema abans de passar a la solució definitiva.

El prototipatge ràpid es refereix a la capacitat de crear un prototipus amb bastant menys esforç del que es necessitaria per produir una implementació operativa definitiva, optimitzada en termes de relació entre cost i prestacions de cara a la seva industrialització.

Un prototipus pot no complir totes les especificacions (temporals, físiques, d'espai,...) que són necessàries per a la versió final del sistema. Per exemple, el prototipus pot satisfer solament un conjunt determinat de les funcions requerides, pot ser expressat en un llenguatge més potent o flexible que la versió final, pot executar-se sobre una màquina amb més recursos que l'arquitectura de la solució final proposada, pot ser menys eficient en temps i en espai que la versió final, pot tenir limitacions de capacitat, pot no incloure les funcions completes de tractament d'errors o pot no tenir el mateix grau de concurrència que la versió final. Totes aquestes simplificacions són sovint introduïdes per fer el prototipus més fàcil i ràpid de construir. El fet que els prototipus permetin integrar diferents aspectes, tots ells de complexitat suficient per haver-se d'implementar per diferents membres de l'equip de desenvolupament, porta a que la metodologia de prototipatge hagi d'estar estretament integrada amb el model organitzatiu d'un determinat projecte, de forma que el prototip pugui ser la plataforma en la qual es poden validar els diferents subsistemes, de forma separada o conjunta.

3.1.1. La necessitat de prototipatge ràpid

Les principals raons per utilitzar prototipus són de tipus econòmic: en la majoria de sistemes és menys car construir prototipus que les versions finals a implementar, tant en termes dels costos de redisseny associats a l'enginyeria no redundant (costos NRE), com a la capacitat de reutilització de les plataformes de prototipatge per a diferents dissenys. Llavors, els prototipus poden ser utilitzats per avaluar els sistemes proposats per tal de ser acceptats pels clients o validar el desenvolupament en cas de dubte. La necessitat de prototipatge és més urgent a causa del creixement de desenvolupament de sistemes més complexos, ja que aquests tenen més probabilitat de tenir errors, i també, són més cars a l'hora de verificar-los i d'implementar-los[11] .

Una altra raó molt important del prototipatge ràpid és que proporciona un mètode eficient i ràpid per fer transferència de tecnologia per desenvolupar nous productes, a través de la estandarditzaci dels components que es poden integrar en diferents sistemes. Això ha donat lloc a una important indústria orientada a la reutilització d'aquests components, també anomenats components virtuals o IP *cores* [12].

3.1.2. El prototipatge com a metodologia

Actualment, el prototipatge tendeix a ser més que un simple conjunt de tècniques disponibles per al disseny de projectes. Aquesta metodologia ha estat formulada per diversos autors com a disseny basat en plataformes (o *Platform-based design*), estructurada com a metodologia convergent al centre ("*meet-in the middle*"), en contraposició a les clàssiques ascendent i descendent. Tot i això, el prototipatge es pot considerar com una metodologia composada per varies tècniques que permeten:

- L'especificació i avaluació del model del sistema a implementar. L'avaluació pot ser feta mitjançant simulació o test. En alguns casos, també es poden aplicar tècniques de verificació formal.
- La captura de l'arquitectura i del funcionament del sistema a desenvolupar.
- La generació de forma automàtica del codi dels components software i/o hardware des del model.
- La integració de components software i/o hardware prèviament verificats, procedents d'entorns externs a l'equip de disseny (components virtuals o IP *cores*).

La capacitat de verificar conjuntament components connectats descrits a diferents nivells d'abstracció (model funcional, descripció HW, descripció SW). De totes maneres, hi ha altres aspectes a considerar en el prototipatge que són:

- La reutilització dels components software. No sempre el components reutilitzats compleixen els requeriments de la nova aplicació.
- L'elecció apropiada del nivell de detall en la descripció del model. Aquesta elecció no és fàcil, ja que una determinada descripció estàndard pel model pot ser incompatible amb certs dominis d'aplicació (sistemes distribuïts, sistemes en temps real, ...).
- La compatibilització entre el model i la verificació formal. Una solució és utilitzar notació formal, però depenent del disseny pot ser impossible, ja que s'ha de tenir uns coneixements elevats en mètodes formals, i les eines i els dominis d'aplicació en els que són vàlids.
- L'automatització de la verificació del model. Les tècniques de verificació no són fàcils de manipular, per tant, haurien de ser automàtiques.

- L'existència d'un entorn amigable com un prerequisit per utilitzar-lo en desenvolupament basat en el prototipatge ràpid. L'entorn s'ha d'adaptar a les necessitats del disseny, i també, s'ha de considerar un aspecte molt important, la compatibilitat amb la plataforma de prototipatge que s'utilitzi.

3.1.3. Prototipatge en sistemes de desenvolupament.

Amb l'objectiu d'ubicar el prototipatge que es presenta en aquesta tesi, es presenta a continuació una classificació segons el tipus de sistema de prototipatge:

- Sistemes de prototipatge virtual. S'utilitzen per a la creació de models software de components a nivell de sistema amb el propòsit de detallar arquitectures i analitzar característiques. Exemples d'aquestes sistemes són Matlab/Simulink i Signal Processing Worksystem (SPW) [19][29].
- Sistemes de prototipatge de HW. Serveixen per emular sistemes i consisteixen en implementar un disseny en una plataforma utilitzant una tecnologia diferent a la tecnologia definitiva. Aquests sistemes es poden basar en només FPGAs o en plataformes que combinen diferents dispositius, tals com FPGAs, microcontroladors, DSP, ... Exemples d'aquestes sistemes són IKOS emulation systems i CoBALT Plus de Cadence [13].
- Sistemes de prototipatge per a plataformes de disseny SoC (*System on Chip*) i NoC (*Network on Chip*). Aquestes plataformes permeten la gestió de *cores* de processadors (ARM, PowerPC, ...), i les seves arquitectures de bus (SoC) o xarxa (NoC), memòria i components perifèrics i/o acceleradors. Exemples d'aquestes sistemes són CEVA-X System1100 i CEVA-X System1200 [14].
- Sistemes de prototipatge per a plataformes de desenvolupament amb FPGAs. S'utilitzen pel desenvolupament de sistemes utilitzant FPGAs. Algunes d'aquestes plataformes poden incloure FPGAs que incloguin *cores* de processadors que permetin el desenvolupament de sistemes SoC. Un exemple d'aquests sistemes són XtremeDSP de Xilinx [17] i DSP Builder d'Altera [18]. També hi ha plataformes específiques per educació, tals com les plataformes de Xess [15].
- Sistemes de prototipatge per a plataformes de computació reconfigurable. S'utilitzen per a sistemes que necessiten computació reconfigurable que bàsicament disposen de processadors, memòries, busos i blocs de computació reconfigurable. Aquests blocs de computació reconfigurable poden ser de gra gruixut o fi. Cal considerar que els sistemes SoC també poden necessitar computació reconfigurable. Exemples d'aquests sistemes són Actel VariCore i Virtex II pro [16].

Segons aquesta classificació, el tipus de prototipatge que es presenta en aquesta tesi s'adapta als sistemes de prototipatge per a plataformes de desenvolupament amb FPGAs. De totes maneres, cal considerar l'opció dels sistemes de prototipatge per a plataformes de computació reconfigurable i dels SoC/NoC, ja que diversos blocs i algorismes de la capa física de l'estàndard HiperLAN/2 i IEEE 802.11a poden ser implementats de forma reconfigurable en sistemes SoC [9] i NoC.

3.2. Requeriments de les plataformes de prototipatge d'un sistema WLAN

La plataforma de prototipatge ve definida per la metodologia, l'arquitectura i la tecnologia que s'utilitzi en el sistema WLAN que es vol desenvolupar. Per tant, primer

de tot, es defineix i s'acota el subsistema WLAN que es vol prototipar que és el processament digital de la capa física segons l'estàndard HiperLAN/2. D'aquesta capa física només es prototipa el mode 3 que correspon a una tassa de transmissió de 12MBits/s amb modulació QSPK, suficient per validar la plataforma de prototipatge.

Metodologia.

És important, per definir una metodologia òptima, conèixer el tipus i les restriccions temporals i computacionals del sistema que es vol desenvolupar. En els apartats 2.2.4 i 2.2.5 s'han exposat aquestes restriccions i es pot extreure que és un sistema de flux de dades continu, on cal destacà que les capes físiques de l'HiperLAN/2 poden suportar tasses de transmissió de fins a 54Mbits/s.

Cal considerar si s'utilitza una metodologia que permeti fluxos de disseny *Top-Down* o *Bottom-Up*. En aquests cas on les restriccions temporals són elevades i considerant que s'utilitzen FPGAs, una metodologia que permeti fluxos de disseny *Bottom-Up* sembla la més adient, ja que permet pujar al nivell de sistema blocs optimitzats a la tecnologia que es vol utilitzar.

Els entorns de disseny a nivell de sistema constitueixen cada vegada més el punt de partida de les metodologies de disseny que inclouen el prototipatge ràpid, ja que disposen de les seves pròpies biblioteques que permeten l'anàlisi de les dades i la seva visualització dels resultats. A més solen suportar diferents llenguatges i normalment permeten treballar amb *cores* IP. Un aspecte important en el prototipatge ràpid es el ventall de possibilitats de verificació que ofereix la metodologia en els diferents nivells d'abstracció inclosos en el seu flux de disseny. La verificació és necessària a cada nivell, però també és molt important la verificació multinivell, tant per qüestions de independència en el desenvolupament dels subsistemes, com per accelerar els temps de verificació.

Arquitectura

La capa física de l'HiperLAN/2 que es vol prototipar disposa d'una arquitectura hardware amb un elevat grau de concurrència. Cal destacar que la part de control que es necessita és mínima i es pot implementar en Hardware. En treballs futurs a aquesta tesi es pot considerar una part software que controli la interfície amb la capa d'accés al medi, així com una implementació HW/SW de la resta de protocols involucrats, utilitzant tecnologies i metodologies estàndard. Per tal que el prototipatge tingui un bon nivell de detall, pot ser interessant disposar d'elements analògics i/o antenes que permetin l'anàlisi a nivell de radiofreqüència.

Tecnologia

S'ha de tenir en compte que per triar la tecnologia cal saber el número i requeriments de les entrades i sortides del sistema, tipus de memòria, restriccions d'alimentació, potència i rellotges. El número d'entrades i sortides de l'emissor i receptor són aproximadament de 40. Les memòries que s'utilitzen bàsicament són: *buffers* per fer sincronitzacions de velocitat, i taules; i bàsicament són de tipus RAM de doble port, RAM i ROM de capacitat petita (128bytes). La gestió de rellotges és important, ja que aquests sistemes WLAN tenen diferents tasses de transmissió de dades dintre d'un mateix flux. Pel que fa als requeriments d'alimentació i energia, aquests solen estar emmarcats per la integració especifica que es realitzi, i els resultats que s'obtenen solen ser escalables a nivell global.

Per suportar la tassa màxima de transmissió de 54MBits/s, la capa física requereix gairebé de 15 GOPS (giga operacions per segon) [3]. Aquestes restriccions computacionals, juntament amb la necessitat de disposar de sistemes de baix consum orientats a solucions de sistemes mòbils alimentats per bateries, porten a la implementació en recursos hardware d'execució concurrent, i com a conseqüència, a

la utilització de FPGAs per al prototipatge d'aquests sistemes, ja que per exemple, les últimes FPGAs Virtex de Xilinx suporten fins a 20 GOPS a causa de les seves arquitectures paral·leles de processament.

3.3. Estat de l'art d'entorns i plataformes per a sistemes WLAN

Al mercat existeixen molts entorns i plataformes de disseny per a sistemes electrònics on la majoria han estat pensats per a un tipus de sistema (sistemes de comunicació, sistemes de control, ...) o tecnologia (ASIC, FPGA, ...) en concret. Llavors cal acotar la recerca, centrant l'estat de l'art en entorns i plataformes que puguin modelar i prototipar sistemes electrònics relacionats amb aquesta tesi. Per tant, es fa un estudi centrat als entorns i plataformes que puguin treballar amb sistemes de comunicació WLAN utilitzant FPGA de Xilinx, i a més, que els entorns tinguin la capacitat de modelar a nivell de sistema.

3.3.1. Entorns a nivell de sistema

Actualment existeixen dues tendències molt clares en entorns per modelar sistemes de comunicació WLAN a nivell de sistema: la utilització d'entorns que utilitzen llenguatges d'alt nivell i d'entorns visuals per a flux de dades.

Llenguatges d'alt nivell

La utilització de SLDLs (*System Level Design Language*) [20][26][33] mostra la seva efectivitat per especificar i verificar models, ja que normalment els estàndards s'especifiquen utilitzant llenguatges de programació, però són poc adequats per a sistemes complexos d'altes prestacions, a causa de la manca d'eficiència que presenten els seus processos d'especificació de concurrència i sincronització de processos, així com de síntesis a l'hora d'obtenir un hardware/software fiable i eficient. Un dels avantatges principals que presenten és la portabilitat tecnològica del codi hardware (i software) que generen.

La majoria d'aquests llenguatges es basen en el llenguatge C. Als llenguatges de programació C/C++ els manca una sintaxi adequada per descriure i controlar eficientment concurrència i sincronització (pròpies tant del hardware com d'arquitectures software i SoC/NoC complexes). Per això, els fabricants d'eines EDA (*Electronic Design Automation*) han desenvolupat en el passat les seves pròpies extensions de C/C++ (Occapi [24], HandelC [25]), la qual cosa havia fragmentat el mercat dels entorns de disseny a nivell de sistema amb capacitat de generar hardware. Per evitar aquesta fragmentació, un nombre important de fabricants van donar suport a la iniciativa OSCI (*Open SystemC Initiative*), la qual utilitza un subconjunt estanditzat del C++ per modelar hardware anomenat SystemC.

SystemC es basa en una plataforma C++ [26] que dóna als enginyers de sistemes i de hardware un entorn de desenvolupament comú per modelar hardware a un nivell alt d'abstracció. Llavors, el model hardware descrit amb systemC pot ser traslladat a un llenguatge HDL sintetitzable per a una determinada arquitectura de hardware.

SystemVerilog és un llenguatge de descripció i verificació hardware (HDVL) [27]. És una extensió del llenguatge Verilog establert en l'IEEE 1364-2001 i a porta millores sobre aquest en la productivitat, readaptabilitat i reutilització .

Entorns visuals per a flux de dades

Els entorns visuals per a flux de dades són idealment adequats per modelar sistemes de processament de dades DSPs, ja que molts dels algorismes s'especifiquen d'una forma més natural a través de gràfics de flux de senyal. Els entorns visuals són similars a les eines tradicionals de captura d'esquemes que

disposen de llibreries de blocs funcionals que permeten composar gràficament el model d'un sistema.

La diferència amb aquestes eines de captura d'esquemàtics, és que els entorns visuals (dels quals en són part els sistemes de prototipatge virtual), contenen a més de les llibreries de blocs funcionals, amb les representacions a diferents nivells, els entorns de verificació (normalment per simulació) integrats en el mateix entorn visual orientats a les diferents aplicacions per al domini de flux de dades (p.e. àudio, vídeo, comunicacions,...). Això permet gestionar de forma eficient els mètodes de verificació juntament amb la gestió i transformació dels diferents tipus i estructures de dades i operadors, que entre altres coses possibilita modelar l'aritmètica específica utilitzant diferents tipus de representacions de dades (enteres, de coma fixa, coma flotant,...) orientats a la implementació del sistema.

Dins d'aquests entorns n'hi ha de dos tipus: els específics per a una determinada tecnologia o fabricant, i els que poden treballar amb diferents tecnologies (ASICs, FPGAs,...). L'avantatge principal dels primers, és la eficiència del hardware que generen i l'inconvenient és la manca de portabilitat a altres tecnologies. En canvi, pels segons és el contrari. D'entre els diferents entorns visuals, en destaquem alguns pels canvis conceptuals han desenvolupat.

El projecte Ptolemy-Ptolemy II [28] es pot considerar un dels referents d'aquests entorns. És un sistema de modelatge i disseny heterogeni que inclou diferents dominis de computació. Inclou una llibreria de components on la majoria d'aquests són de dominis polimòrfics, per tant, poden funcionar en varis dominis. Això es deu a la propietat d'ortogonalitat de les estructures de dades flexibles, respecte de les arquitectures i entorns d'aplicació.

Signal Processing Worksystem (SPW) de Coware [29] (anteriorment de Cadence), COSSAP de Synopsys [30], DSP-Station de Mentor Graphics [31] i Advanced Design System (ADS) d'Agilent Technologies [32] són entorns visuals comercials per a aplicacions de sistemes de flux de dades que s'utilitzen pel disseny de sistemes DSP. Aquests entorns disposen de biblioteques amb components d'un alt nivell d'abstracció. També inclouen biblioteques que contenen blocs sintetitzables pels diferents fabricants de FPGAs. Tots aquest entorns són molt potents, però el seu cost es molt elevat en comparació a entorns comercials específics per a FPGAs [17][18].

Simulink és l'entorn gràfic de Matlab [19], un entorn molt popular de MathWorks per al modelatge i verificació d'algorismes que es resolen típicament de manera matricial. Disposa de diferents biblioteques per modelar sistemes DSP i sistemes de comunicacions amb una gran capacitat d'anàlisis de dades i de visualització de resultats. Simulink és una excel·lent plataforma per a eines de disseny de sistemes encastats ja que disposa de mòduls específics (*toolboxes*) per a generació de codi SW (procesadors digitals de senyals) i HW (FPGAs).

System Generator és una eina de Xilinx [17] específica per al disseny de sistemes DSP per a les seves FPGAs. Aquesta eina s'integra sobre la plataforma Matlab/Simulink formant un entorn visual per a flux de dades, el qual permet un nivell alt d'abstracció visual del sistema a modelar. A més, proporciona una llibreria de blocs per l'entorn de Matlab/Simulink que facilita la interacció amb el seus diferents paràmetres, permetent incorporar-hi expressions matemàtiques de Matlab. Cal destacar, la capacitat de poder combinar els dissenys realitzats amb components virtuals (IP) o entitats descrites pel dissenyador, directament utilitzant llenguatges HDL (VHDL o Verilog). System Generator trasllada el model dissenyat a través de un procés automàtic a un hardware eficient sintetitzable (VHDL o Verilog), creant tots els fitxers necessaris per produir un projecte que s'integri directament a l'eina de síntesis física lògica i sobre les seves FPGAs (ISE). La implementació és eficient a través de la utilització de cores IP que proporcionen un rang de funcionalitat des d'operacions

aritmètiques bàsiques fins algorismes complexos per a DSP. També, existeix la possibilitat de crear un fitxer *testbench* per a l'eina de simulació de VHDL o Verilog a partir dels estímuls aplicats als models des de Matlab/Simulink, i així com interaccionar amb plataformes HW des de Matlab, a nivell d'enviar-hi estímuls i recollir-hi resultats per processar-los. Les jerarquies del codi VHDL generat mantenen les mateixes jerarquies dels diferents sistemes i subsistemes que tenia el model.

DSP Builder és una eina equivalent d'Altera [18] sobre la plataforma de Matlab/Simulink i pensada per a les seves FPGAs. Les prestacions són molt semblants a les que ofereix System Generator de Xilinx.

3.3.2. Estat de l'art de les plataformes orientades als sistemes WLAN

Les plataformes que interessen són les pensades específicament per a sistemes DSP que incorporen FPGA Virtex II de Xilinx. Segons la complexitat del model del sistema HiperLAN/2 presentada en l'apartat 2.2.5 s'ha fet una estimació màxima inicial d'ocupació, que ens porta a utilitzar FPGAs de com a mínim 2 milions de portes i, a més, que disposin convertidors AD (Analògic/Digital) i DA (Digital/Analògic) d'alta velocitat. Un altre aspecte a tenir en compte és que s'adapti el més possible a l'entorn format per Matlab/Simulink, System Generator i ISE.

Segons aquestes característiques hi ha diverses plataformes en el mercat actualment que compleixen aquest requisits.

El fabricant Alpha Data [35] disposa d'una plataforma ADM-XCR-II que incorpora una FPGA Virtex II de Xilinx (2V3000, 2V4000, 2V6000, 2V8000), 6 bancs de memòria independents ZBT SRAM, bus PCI (66MHz 64-bitI), bus propi XCR, 256MBytes de memòria dinàmica DRAM i 16 MByte de memòria Flash. A més en funció del tipus d'aplicació, per exemple per a un sistema WLAN, s'hi pot adaptar una placa XRM ADC o XRM DAC que disposen de convertidors AD i DA d'alta velocitat.

El fabricant Annapolis MicroSystems [36] disposa d'una plataforma PCcard Wildcard II que incorpora una FPGA Virtex II de Xilinx (XC2V3000), 2 Mbytes de SRAM ZBT, 64 MByte de memòria dinàmica DRAM i un ADC de 12 bits a 65MHz. També disposen de la plataforma Wildstart 2 que pot incorporar una o dues FPGAs Virtex II de Xilinx (XC2V6000 o XC2V8000). A més inclou 12 bancs de memòria SRAM (48 MBytes), 2 bancs de memòria dinàmica SDRAM (256MBytes), memòria Flash i un Bus PCI (32/64 bits,33/66 MHz). Per aplicacions WLAN s'hi pot incorporar plaques filles que disposen de convertidors DA i AD d'alta velocitat.

El fabricant Lyrtech [37] disposa de varies plataformes que s'adeqüen. Les més interessants són la plataforma Signal Wave i la Signal Master. La Signal Wave incorpora una DSP de Texas Instruments (TMS320C6713), una FPGA Virtex II de Xilinx (XC2V3000), memòria SRAM (32MBytes) per a la FPGA, memòria SDRAM (32MBytes) per a la DSP i convertidors AD i DA a 65MHz. La Signal Master incorpora una DSP de Texas Instruments (C6701), una FPGA Virtex 2 de Xilinx (fins la XC2V8000), memòria SDRAM (16 a128MBytes) per a la FPGA, memòria SDRAM (16MBytes) i SBSRAM (fins a 2Mbytes) per a la DSP. A més s'hi poden acoblar plaques filles que incorporen convertidors AD i DA d'alta velocitat.

El fabricant Nallatech [34] disposa d'una plataforma formada per una placa mare BenONE i una placa filla BenADDA. La plataforma inclou una FPGA Virtex II de Xilinx (XC2V2000, XC2V3000, XC2V6000), Bus PCI (64bit/33MHz PCI, 32bit/33MHz PCI), Memòria SRAM ZBT (8Mbytes), 2x 14-Bit ADC i 2x 14-Bit DAC. Aquesta plataforma s'inclou en el Kit de desenvolupament XtremeDSP de Nallatech que està format bàsicament per les eines System Generator i ISE de Xilinx, Fuse de Nallatech i Matlab/Simulink de Mathworks.

3.4. Dispositius lògics programables: FPGA

Programmable Gate Arrays) i CPLDs (*Complex Programmable Logic Devices*). En principi la diferència d'arquitectures entre aquests dispositiu era molt clara i originada en la procedència de la estructura de repetició en cel·les de multiplexors, els primers, o en arrays matricials (tipus PAL/PLA/PLD) els segons. Tot i això, els interessos comercials associats al nom de FPGA (registrat per Xilinx), han fet que aquestes diferencies d'arquitectures no siguin tan evidents. També existeixen altres noms per referir-se a aquests dispositius com FPDs (*Field Programmable Devices*) o HCPLDs (*High Complex Programmable Logic Devices*). De totes formes el nom més popular per als dispositius reconfigurable complexos és el nom de FPGA.

Les FPGAs, en general, són composades d'una sèrie de blocs lògics configurables (CLB), de blocs d'entrada/sortida i de connexions programables amb la utilitat d'implementar un circuit digital (tot i que també n'hi comença a haver d'analògiques, encara són poc significatives). La Figura 3.1 mostra l'arquitectura d'un model genèric d'una FPGA.

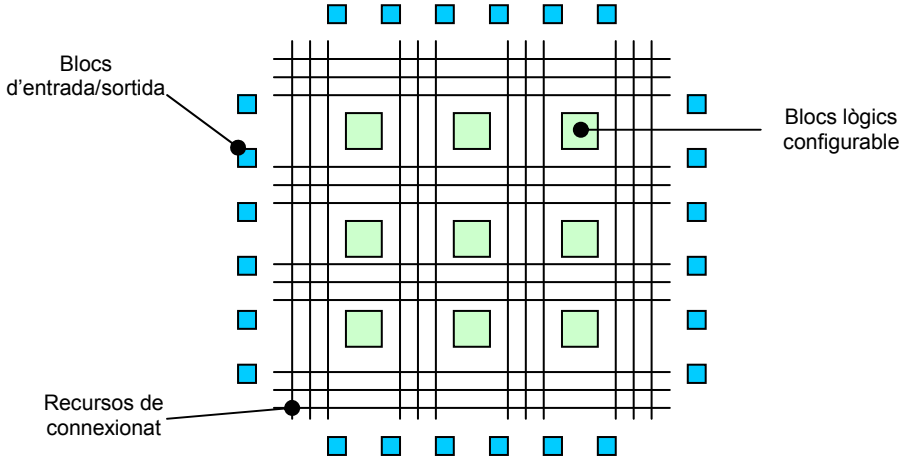


Figura 3.1 Arquitectura d'un model genèric d'una FPGA

Els blocs lògics normalment contenen una o més cel·les per implementar funcions combinacionals, i un o més elements de memòria (registres d'un bit) per guardar resultats parcials i/o implementar circuits seqüencials. Les parts combinacionals es modelen com a LUTs (*Look Up Tables*), producte de termes (combinacions de portes AND i OR), portes XOR, multiplexors o en algun cas components més específics.

Els blocs d'entrada/sortida permeten definir la funció especifica de cada pin (entrada, sortida, bidireccionals), les seves propietats de sistema (E/S registrades, scan test), elèctriques (*pull-up*,...), i de tipus de connectivitat amb diferents estàndards i terminacions (CMOS, TTL, LVDS,...).

Els blocs de connexions permeten diferents nivells *fan-out*, velocitat de transmissió (en FPGAs de prestacions elevades), etc.

Pel que fa a la tecnologia de configuració, existeixen diferents dispositius que en donen les propietats, essent les bàsiques si la configuració és volàtil (p.e. SRAM), o no-volàtil (p.e. Flash, EEPROM, fusible i antifusible) i que s'apliquen tant a les funcions implementades així com a les interconnexions.

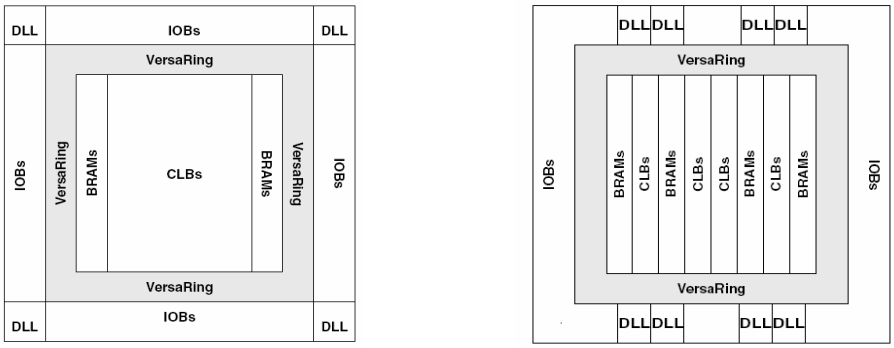
3.4.1. Família Virtex

La família de FPGAs Virtex ha anat evolucionant des de les FPGAs Virtex fins a les Virtex II Pro. Cal destacar d'una banda, l'evolució causada per les variacions tecnològiques que ha sofert la família Virtex i que afecta les propietats globals segons

els mecanismes d'escalat clàssics (en freqüència, tensió d'alimentació, consum, número de LUTs,...), i d'altra banda cal destacar principalment la variació de la seva arquitectura, amb la incorporació de més recursos disponibles i més dedicats a funcions específiques.

La FPGA Virtex disposa d'una estructura jeràrquica en la qual l'element de repetició és un conjunt central de CLBs que incorpora en els dos costats una columna de blocs de memòria dedicada RAM. Entre mig de la matriu d'aquests elements i els blocs d'entrada/sortida hi ha un anell de connexions que permet la seva connexió anomenat *VersaRing*. També incorpora una distribució complexa de senyals de rellotge, que pot estar per exemple composada de 4 DLLs (*Delay-Locked Loop*) als extrems de la FPGA (Figura 3.2a).

La FPGA Virtex E disposa d'una estructura central de CLB similar però incorpora més memòria dedicada distribuïda en forma de columna entre els CLBs. Igual que la família Virtex disposa d'un anell de connexions *VersaRing*, però inclou 8 DLLs situats a dalt i a baix de la FPGA (Figura 3.2b).



a) b)

Figura 3.2 Arquitectures Virtex: a) Virtex, b) Virtex E

La FPGA Virtex II té una estructura central bastant diferent a les dues famílies anteriors (Figura 3.3a) ja que entre les formacions de CLBs, s'incorporen columnes de memòria RAM i de multiplicadors. En aquesta arquitectura desapareix l'anell de connexions *VersaRing* i es passa amb un sistema de connexions anomenat *Active Interconnect Technology*. Aquest sistema es basa en que cadascun dels elements configurables disposa d'una matriu de connexions que permet la connexió a una matriu global de connexions. També es canvien els DLLs per gestors digitals de rellotge DCM[1] (*Digital Clock Manager*).

La FPGA Virtex II Pro torna a canviar d'estructura incorporant *cores* encastats (en forma de silici no reconfigurable) del microprocessador *PowerPc 405* i del *Rocket I/0 serial transceiver* (Figura 3.3b). El processador és de tipus RISC de 32 bits i el *Rocket I/0 serial transceiver* és un transmissor flexible de paral·lel a sèrie i de sèrie a paral·lel d'alta velocitat, utilitzat per interconnexions de busos d'una amplada de banda alta.

[1] El gestor digital de rellotge DCM permet generar nous rellotges (interns o externs a la FPGA) en fase amb el rellotge d'entrada, sintetitzar un rang ampli de freqüències de sortida (multiplicant o dividint el rellotge d'entrada) i controlar el desplaçament de fase del rellotges de forma dinàmica .

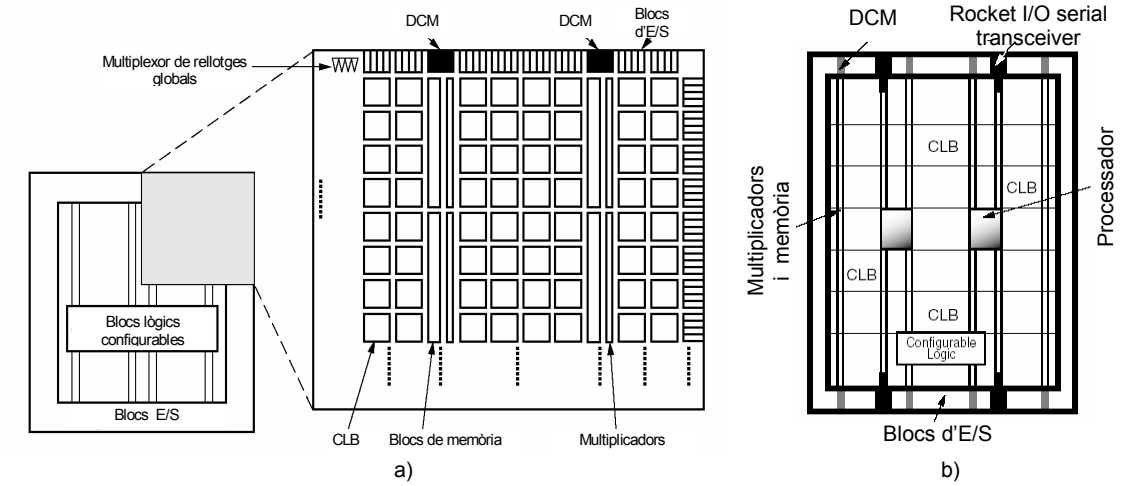


Figura 3.3 Arquitectures Virtex: a) Virtex II, b) Virtex II Pro

3.4.2. Arquitectura de la Virtex II

Bàsicament en aquest apartat donem una idea de l'arquitectura de les FPGAs avançades de Xilinx, en concret de la família Virtex II, per ajudar el lector a interpretar millor la terminologia utilitzada en els pròxims capítols.

La família Virtex II és una plataforma FPGA de desenvolupament d'altes prestacions que permet treballar amb *cores* IP o mòduls personalitzats. La família ofereix solucions completes per a aplicacions de telecomunicacions sense fils (*wireless*), imatge i processament del senyal. També inclou interfícies orientades a aquestes aplicacions com poden ser PCI, LVDS i DDR. La tecnologia de fabricació utilitzada és CMOS 0.15/0.12µm amb 8 capes de metall i la seva tecnologia de programació és SRAM (no-volàtil). L'arquitectura de la Virtex II s'ha optimitzat per a velocitats de treball altes i per baix consum.

La Figura 3.4 mostra una vista general de l'arquitectura d'una FPGA Virtex II. La principal diferència entre aquesta arquitectura i la presentada en la Figura 3.1, és que aquesta inclou blocs de memòria RAM, multiplicadors i gestors de rellotges (DCM).

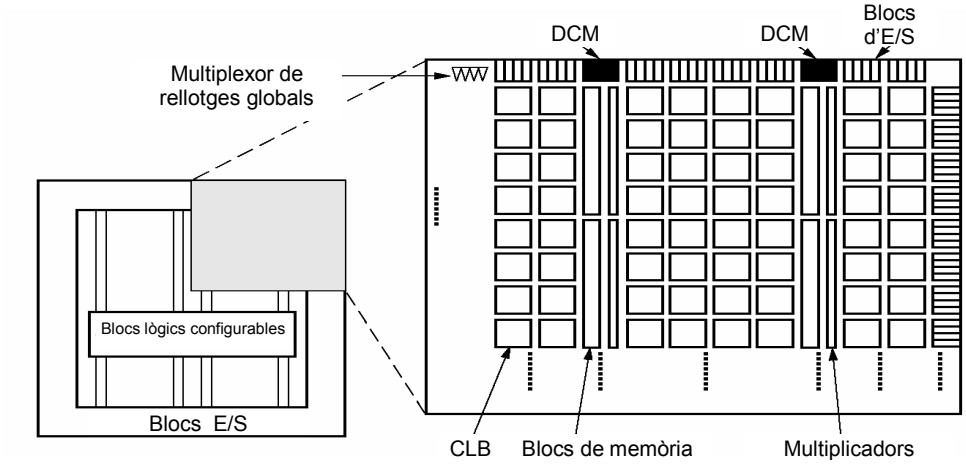


Figura 3.4 Arquitectura general d'una Virtex II

Cal destacar de l'arquitectura lògica configurable els següents elements: els blocs lògics configurables que disposen d'elements combinacionals i seqüencials, els blocs de memòria RAM *dual-port* de 18Kbit, els blocs multiplicadors dedicats de 18x18bits i els blocs de gestió de rellotge (DCM) que ofereixen diverses funcions de multiplicació, divisió i de compensació del retard de distribució del rellotge.

Els recursos de connexió, anomenats *Active Interconnect Technology*, interconnecten tots aquest elements. La matriu de connexió general (GRM) és una formació d'interruptors de connexió. Llavors, cadascun dels elements programables esta lligat a una matriu de connexió permetent múltiples connexions a la matriu de connexió general.

Altres aspectes a destacar fan referència a l'arquitectura del blocs d'entrada/sortida i dels blocs lògics configurables. Els blocs d'entrada i sortida s'estructuren en tres categories (Figura 3.5a): bloc d'entrada amb un registre opcional *single-data-rate* (SDR) o *double-data-rate* (DDR), bloc de sortida amb registre opcional SDR o DDR i un *buffer* opcional amb opció de nivell d'alta impedància (*3-state*) activat directament o a través d'un registre SDR o DDR i Bloc bidireccional. També cal afegir que els blocs d'entrada i sortida suporten diferents estàndards per senyals d'un pin/bit (LVTTL, LVCMOS, PCI-X, PCI, Cardbus, ...) i diferencials (LVDS, BLVDS, ULVDS, ...). Alguns d'aquests estàndards necessiten d'una font de tensió externa addicional. Llavors, aquests voltatges es connecten al dispositiu en diferents grups de blocs de pins d'entrada i sortida anomenats bancs. La FPGA es divideix en 8 d'aquests bancs (Figura 3.5b).

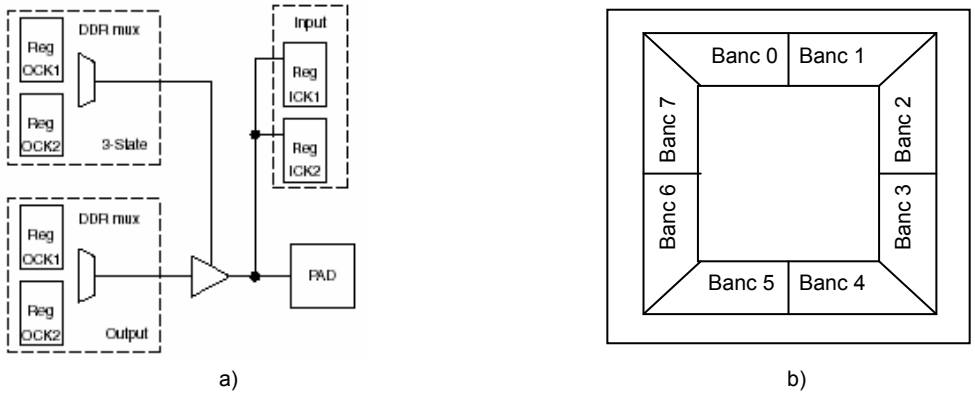


Figura 3.5 a) Bloc d'entrada/sortida, b) Distribució de bancs d'una FPGA Virtex II

Els recursos d'un bloc lògic configurable inclouen; (i) una matriu de connexió, connexions locals ràpides entre veïns, (ii) 4 *slices,* i (iii) dos *buffers* 3-estats (Figura 3.6a). Un *slice* és el bloc bàsic que conté l'arquitectura lògica configurable d'un CLB. Llavors cadascun dels *slices* està format per: dos generadors de funcions de quatre entrades, una cadena lògica de *carry*, portes lògiques aritmètiques de control de polaritats, dos elements de memòria (amb funció de registre/*latch*) i multiplexors. Cal destacar que els dos generadors de funcions es poden configurar com a LUTs de quatre entrades, registre de desplaçament de 16 bits o memòria distribuïda RAM de 16 bits (Figura 3.6b).

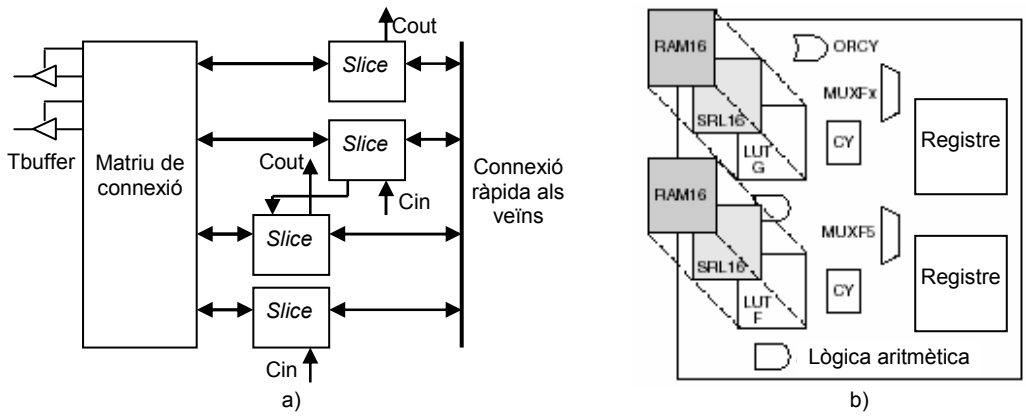


Figura 3.6 a) Bloc lògic configurable (CLB), b) Configuració d'un *slice*

3.5. Metodologia de prototipatge d'un sistema WLAN

En aquesta tesi es proposa una metodologia de prototipatge ràpid que vagi des de Matlab/Simulink a la implementació final del sistema en una FPGA de Xilinx. S'ha decidit treballar amb l'entorn visual comercial format per Matlab/Simulink i System Generator per a FPGAs de Xilinx, conjuntament amb l'entorn propi (ISE) de Xilinx de síntesi i de *place & route*. També, s'ha incorporat el *kit* de desenvolupament XtremeDSP de Xilinx que inclou la plataforma de Nallatech BenOne/BenADDA [34] amb la seva l'eina de programació i control FUSE. Cal remarcar que aquesta metodologia permet fluxos de disseny *top-dow*n o *bottom-up* però bàsicament per aquests sistemes es treballarà en *bottom-up*. Cal considerar que aquesta metodologia també és transportable a les FPGAs d'Altera, només caldria utilitzar les seves eines (DSP builder, Quartus II, Signal Tap). A continuació s'exposen el mètodes de sincronització, fluxos de disseny, verificació i quantificació d'aquesta metodologia.

3.5.1. Mètodes de sincronització

Un dels aspectes més importants en el disseny del subsistema (capa física) Hiperlan/2 són els mètodes de sincronització, ja que aquests subsistemes presenten diferents tasses de transmissió de dades dins del flux continu d'informació i, per tant, necessitat de sincronisme. Llavors, dins del domini computacional orientat al model de flux de dades, existeixen diferents mecanismes de control (dades, sincronisme) en més o menys nivell d'automatització:

- Sincronisme de dades: blocs polimòrfics que dedueixen el tipus i el número de bits de la sortida en funció del tipus i el número de bits de l'entrada. El dissenyador pot decidir que aquest mecanisme sigui automàtic o no.
- Senyals de validació (mecanisme de sincronització): senyals que permeten la sincronització dels diferents blocs síncrons que constitueixen el flux de dades. Els blocs més complexos de Xilinx incorporen aquests senyals de validació formats per un bit d'entrada i un de sortida. El bit d'entrada informa al bloc en quin moment les dades són vàlides, i el bit de sortida, informa al pròxim bloc quan les dades són vàlides. La resta de blocs no incorporen aquests bits, per tant, en cas de ser necessari s'hauran d'incorporar.
- Compartició de memòria: usualment utilitzant memòries RAM de doble port completament asíncrones, que permeten la gestió de dades pels dos ports a diferents velocitats. Aquests tipus de dispositius permeten la sincronització de diferents velocitats en un mateix flux de dades, típicament unidireccional.
- Controls cablejats utilitzant màquines d'estats finits (FSM) senzilles del tipus Mealy o Moore. Aquests blocs permeten fer controls senzills dels diferents blocs que formen el flux de dades. Per exemple el control de l'algorisme IFFT/FFT.
- Controls programables mitjançant la inserció de controladors programables (p.e. picoblaze dins Xilinx). Aquest petit microcontrolador permet fer controls més complexos que inclouen petits càlculs (amb unitats funcionals molt senzilles).
- Gestió de sistemes amb diferents tasses de transmissió de dades: blocs que permeten pujar o baixar una relació entera del període del rellotge en qualsevol punt del flux de dades.
- Gestió de diferents dominis de rellotge: possibilitat de treballar amb diferents dominis de rellotge en un mateix flux de dades.

3.5.2. Flux de disseny

En la metodologia de prototipatge que es presenta podem treballar amb dos fluxos de disseny: un flux de disseny bàsic i un flux de disseny amb coverificació.

a) Flux de disseny bàsic

El flux de disseny bàsic permet treballar amb diferents plataformes i disposa de varis passos de refinament, verificació a cada nivell i multinivell, tal com es mostra a la Figura 3.7.

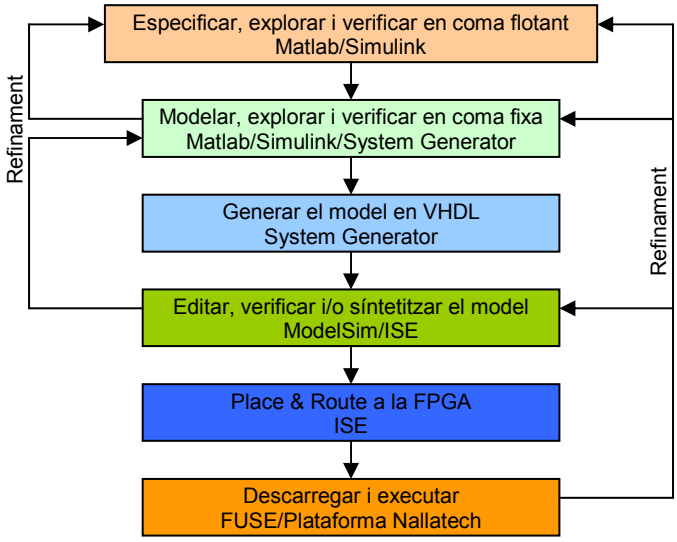


Figura 3.7 Flux de disseny bàsic

A continuació s'expliquen els diferents passos que formen el primer flux de disseny de la Figura 3.7.

En funció de les especificacions del sistema, s'identifiquen els algorismes de processament adequats i es procedeix a la simulació amb el llenguatge Matlab/simulink sobre variables en coma flotant. Aquest procés no sempre és necessari i, per tant, en aquest casos, es passa directament a l'especificació i exploració del model en coma fixa. A vegades s'especifica un model equivalent en coma flotant per fer la comparació amb el model de coma fixa.

Per a la verificació, s'utilitza l'entorn de simulació de Matlab/Simulink, el qual disposa de diferents biblioteques (*toolboxes*), on cal destacar les de comunicació i processament del senyal digital que aporten molts algorismes ja especificats. Aquests algorismes conjuntament amb les diferents alternatives d'aplicació d'estímuls, anàlisis de dades i visualitzadors, permeten una verificació molt eficient i ràpida.

Explorar i verificar el model en coma fixa.

El pas del model descrit en coma flotant a coma fixa en un procés manual. Aquest procés consisteix en substituir o adaptar les diferents parts de l'algorisme en coma flotant per blocs funcionals de coma fixa (que subministra el fabricant de FPGA) d'acord amb el compromís cost-prestacions requerit. En cas que l'algorisme no estigui especificat en coma flotant, es pot especificar, explorar i verificar el model en coma fixa, amb l'ajuda dels diferents blocs d'anàlisi de dades i visualitzadors que disposa Matlab/Simulink. També, si en el model de coma fixa hi ha algun bloc HDL, es pot verificar el model utilitzant la cosimulació o simulació concurrent de models algorísmics (matlab) i models de descripció del hardware (VHDL), mitjançant els simuladors integrats en la metodologia (p.e. Matlab/Simulink i ModelSim).

La realització de sistemes de processament en coma fixa, ja sigui sobre hardware específic o programable, requereix la correcta configuració de les longituds de paraula de tots els senyals. La monitorització del marge dinàmic i el número de bits de les variables és de vital importància en el desenvolupament de sistemes de processament en coma fixa.

Generar el model en VHDL.

El model obtingut en coma fixa es passa a un llenguatge de descripció de hardware, en aquest cas, VHDL, d'una forma automàtica mitjançant una de les funcions de l'eina System Generator. Cal especificar prèviament a l'eina el tipus de FPGA i la velocitat del seu rellotge per a la creació del fitxer de restriccions físiques i temporals. En aquest procés, apart de generar el codi VHDL dels diferents blocs bàsics de Xilinx, l'eina System Generator pot fer diverses crides a l'eina Core Generator, per tal que generi els *cores* de Xilinx que hi hagi en els blocs del model. Al final d'aquest procés s'obté un codi VHDL fiable i sintetitzable (i eventualment òptim) i també, un projecte amb tots els fitxers necessaris per a la seva eina de síntesis ISE.

Editar, verificar i/o sintetitzar el model hardware.

En aquest pas, el model VHDL es pot editar amb l'entorn ISE o verificar amb el simulador de HDL ModelSim. Per exemple, es podria editar per inserir DCM (*Digital Clock Manager*) per fer retocs en la gestió de rellotges. Per fer la síntesi s'utilitza l'eina ISE de Xilinx. Normalment, l'edició i simulació del codi VHDL no es sol fer i es passa directament a la síntesi del codi. Per fer la síntesi s'utilitza l'entorn de Xilinx ISE.

Place & Route a la FPGA.

Un cop el codi VHDL s'ha sintetitzat, es procedeix al *place & route* amb l'entorn ISE. Si el model té unes restriccions temporals molt exigents, cal augmentar el nivell d'esforç de *place & route* a través de les propietats d'aquest apartat. Si no es compleixen les restriccions temporals, s'haurà de repetir aquest procés augmentat un grau més l'esforç de *place & route*. En cas que no s'aconsegueixi complir les restriccions temporals, caldrà analitzar els informes que genera l'eina, per tal d'identificar el punt feble i tornar a refinar el procés des dels diferents camins que ofereix el flux de disseny.

Descarregar i executar.

Per descarregar el model a la FPGA, primer s'ha de generar el fitxer de configuració mitjançant l'entorn ISE de Xilinx. Un cop generat aquest fitxer, es descarrega a la FPGA a través de l'eina FUSE de Nallatech i llavors, s'executa el model sobre la plataforma. Es pot comprovar el seu funcionament utilitzant un analitzador lògic intern/extern i/o un analitzador d'espectre. Cal remarcar que si es disposa del cable adequat pel bus JTAG es pot fer la descarrega del fitxer a través del mateix entorn ISE.

b) Flux de disseny amb coverificació

Una alternativa al flux de disseny anterior, és el flux de disseny que es mostra a la Figura 3.8. El flux de disseny amb coverficació és específic per una determinada plataforma, i respecte del presentat anteriorment, automatitza diversos passos mercès al perfilat d'una determinada plataforma de prototipatge a nivell de Matlab (tipus de FPGA, entrades i sortides, convertidors D/A i A/D, entrades de rellotge, ..) per crear un model de coverificació hardware. En aquest cas s'ha utilitzat un perfil de plataforma per a sistemes DSP, la qual s'adapta molt bé al disseny de subsistemes WLAN. És un flux de disseny amb major automatització (i per tant productivitat del disseny), però amb l'inconvenient d'estar molt més lligat a la plataforma que s'ha perfilat. En cas de

voler treballar amb una altra plataforma amb el mateix flux de disseny, cal crear un perfil nou adaptat a la plataforma.

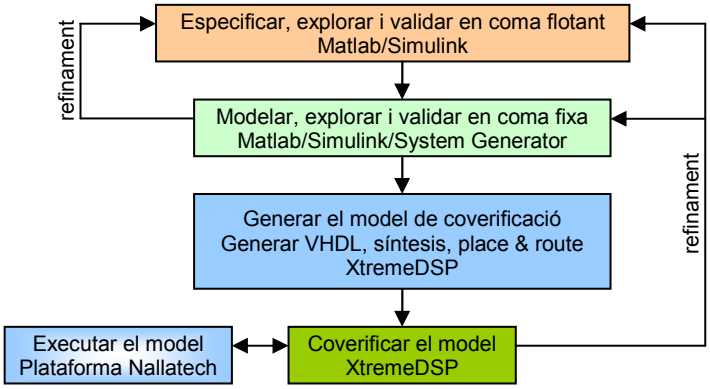


Figura 3.8 Flux de disseny amb coverificació

Vegem a continuació els diferents passos que segueix aquest flux de disseny:

Els dos primers passos d'especificar, explorar i validar en coma flotant i coma fixa són idèntics que el flux de disseny anterior.

Generar el model de coverificació.

Aquest flux permet que, després d'obtenir el model en coma fixa, es pugui generar el codi VHDL, fer la síntesi i *place & route* conjuntament d'una forma automàtica. Cal especificar prèviament el perfil de la plataforma i la velocitat del rellotge de la FPGA, per tal de crear el fitxer de restriccions físiques i temporals. El resultat d'aquest procés és un model de coverificació.

Coverificar el model.

El model de coverificació obtingut es pot utilitzar per verificar la funcionalitat del model hardware dins l'entorn de Simulink. Normalment, es substitueix el model en coma fixa pel model de coverificació deixant igual tots els estímuls i visualitzadors que hi havia en el model de simulació en coma fixa. Durant la simulació, el bloc hardware de coverificació interacciona amb la plataforma de la FPGA, automatitzant les tasques de configuració de la FPGA, transformació i transferència de dades (estímuls i respostes), i de programació del rellotge. Aquests blocs consumeixen i produeixen el mateix tipus de senyals que un altre bloc de l'entorn que s'estigui utilitzant, i per tant es poden utilitzar diferents mecanismes de verificació (equivalència, qualitatius, quantitatius, etc.)

3.5.3. Verificació heterogènia multinivell

Els fluxos de disseny presentats ofereixen diferents alternatives de verificació: verificació a cada nivell i verificació heterogènia multinivell. En aquest cas, s'entén per verificació heterogènia multinivell, les verificacions entre diferents passos del flux de disseny o de dominis (hardware o software). Les possibilitats de simulació multinivell que s'ofereixen entre els dos fluxos de disseny presentats són: verificació mitjançant un analitzador lògic, verificació amb la confrontació de models de coma fixa i flotant, coverificació i cosimulació.

a) Verificació mitjançant un analitzador lògic

La Figura 3.9 mostra el flux de disseny utilitzant la verificació multinivell integrant una analitzador lògic en el disseny. Aquesta verificació permet la verificació funcional i temporal del disseny treballant sobra la plataforma.

La verificació consisteix en integrar un analitzador lògic al disseny, des de l'entorn Matlab/Simulink, mitjançant un bloc de la biblioteca de System Generator. Aquest bloc permet fer les connexions oportunes a les dades que es volen analitzar conjuntament amb les condicions de captura que serviran per donar les ordres de captura i emmagatzemament de les dades seleccionades dins la memòria RAM no utilitzada de la FPGA en condicions de temps real. A continuació es segueix el flux de disseny. Un cop el model s'executa sobre la plataforma, s'activa l'analitzador lògic (eina Chiscope de Xilinx) [17] que permet connectar-se a la plataforma, sempre i quan s'hagi integrat l'analitzador lògic en el disseny.

D'aquesta forma, es permet visualitzar els resultats i passar-los a l'entorn Matlab/Simulink, per tal de fer-ne les anàlisis necessàries. Cal remarcar que la captura dels resultats a través de l'eina Chiscope és en temps real, i es fa cada vegada que es compleixen les condicions de captura imposades, ja que l'analitzador lògic s'executa sobre la plataforma. En canvi, el pas de passar els resultats del Chipscope a Matlab/Simulink no es en temps real.

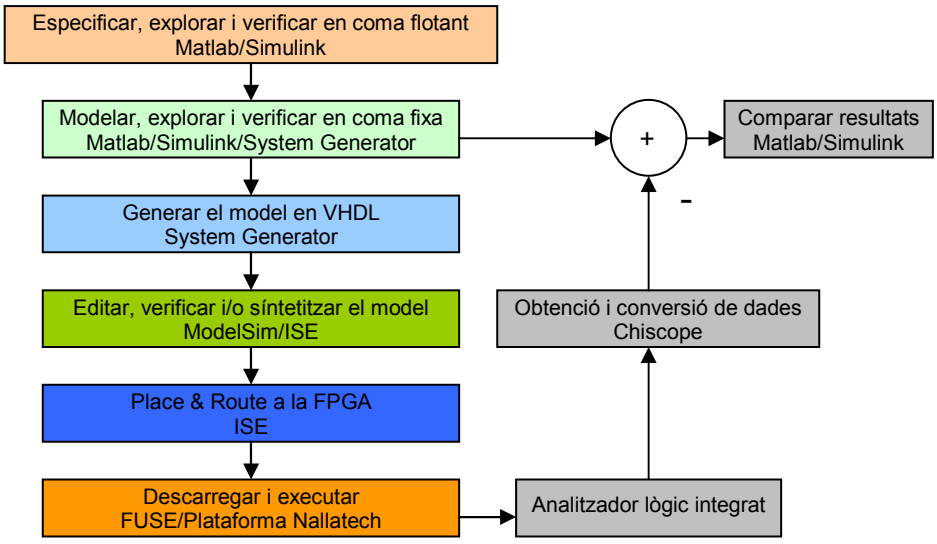


Figura 3.9 Verificació multinivell mitjançant un analitzador lògic integrat

b) Verificació amb la confrontació de models de coma fixa i flotant

Aquesta verificació consisteix en comparar un model de coma flotant igual o equivalent a un model en coma fixa. La Figura 3.10 mostra els dos passos del flux del disseny on s'aplica aquesta tècnica de verificació. Aquesta tècnica permet explorar i verificar la funcionalitat de l'algorisme o model que s'estigui dissenyant d'una forma molt ràpida, ja que bastants algorismes que s'apliquen als sistemes WLAN són inclosos en les llibreries de Simulink.

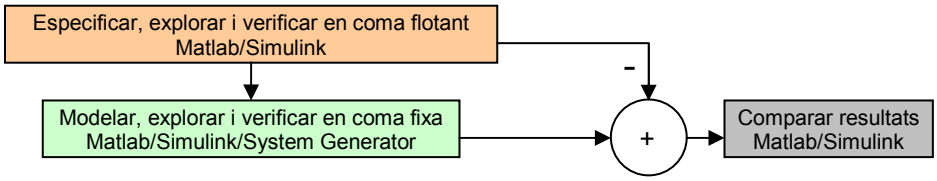


Figura 3.10 Verificació a través de confrontació de models de coma fixa i flotant

c) Coverificació

La Figura 3.11 mostra el flux de disseny amb coverificació. En aquesta tesi s'utilitza el terme coverificació per referir-se a una simulació conjunta del simulador Simulink i un hardware que corre sobre la plataforma, cridat a través d'un model de coverificació

prèviament generat. Llavors, la coverificació permet la verificació funcional del hardware. Normalment, aquesta verificació no és en temps real.

Durant la simulació, el model de coverificació interacciona amb la plataforma de la FPGA, automatitzant les tasques de configuració de la FPGA, transferència de dades i de programació del rellotge. Aquests blocs consumeixen i produeixen el mateix tipus de senyals que un altre bloc de l'entorn que s'estigui utilitzant. Llavors, quan un valor és escrit a un dels ports d'entrada del bloc, aquest envia la corresponent dada al hardware corresponent. D'una forma similar el bloc rep dades des del hardware quan hi ha un event en un sortida del bloc.

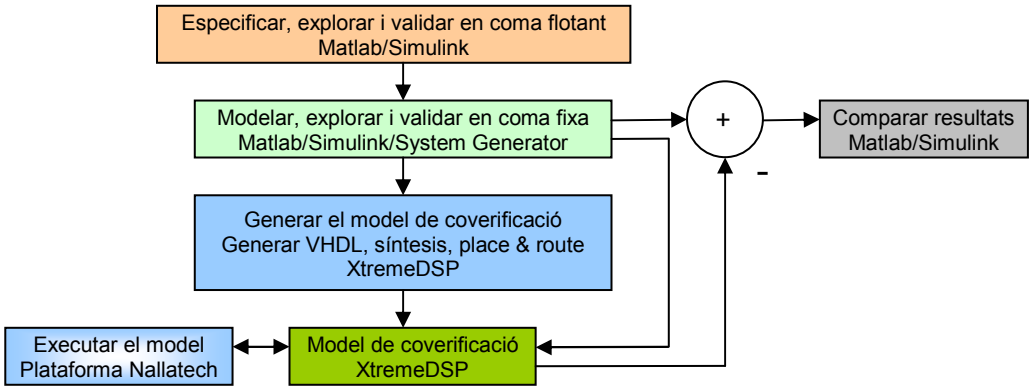


Figura 3.11 Verificació multinivell mitjançant un model de coverificació

Hi ha diferents camins per sincronitzar el model de coverificació amb la plataforma de la FPGA durant les simulacions. El tipus de mode de funcionament del rellotge que es selecciona a través del bloc determina aquesta sincronització. Hi ha dos modes de funcionament a triar: *single-step* i *free-running*.

- Mode *single-step*: hi ha una llaç tancat pas a pas entre el hardware i el software. Això s'aconsegueix donant un sol cicle de rellotge (o varis cicles de rellotge si el disseny utilitza diferents tasses de transmissió de dades) al hardware per a cada cicle de simulació. Aquest mode permet la verificació funcional però no treballa en temps real.

- Mode *free-running*: el hardware treballa de forma asíncrona respecte de la simulació software. A més, el rellotge del hardware funciona contínuament, la qual cosa és consistent amb el model de sistemes de flux de dades. Utilitzant aquest mode el funcionament del bloc hardware de coverificació canvia, ja que els ports d'entrada i sortida estant molt temps sense sincronitzar-se amb els events de Simulink. Quan passa un event en un port de Simulink, tenim una latència, associada al bus de comunicació amb la placa de prototipatge (p.e. USB), des que el valor es llegeix o s'escriu al corresponent port en hardware. Llavors, en aquest període hauran passat un número desconegut de cicles de rellotge de hardware que fan impossible saber-ne el seu estat actual. De totes maneres, es poden implementar mecanismes explícits de sincronització dins del model, per treballar en un model equivalent al de temps real. Un exemple possible, seria incloure un registre d'estat que s'actualitzi quan es reuneixin una sèrie de condicions establertes, el qual es pugui llegir per determinar l'estat que es troba el hardware.

d) Cosimulació

La cosimulació permet la verificació conjunta de blocs Simulink/System Generator i de blocs descrits en llenguatges de descripció de hardware (VHDL, EDIF), en el pas del flux de disseny on es modela i s'explora en coma fixa. Per fer possible aquesta verificació és necessària la sincronització entre dos simuladors: Simulink i ModelSim. Simulink per fer la simulació dels blocs propis i dels de System Generator. ModelSim

per fer la simulació dels blocs descrits de VHDL o EDIF. El principal problema que presenta aquesta tècnica de verificació és que les simulacions són molt lentes, per procés propi de sincronització d'events. Per tant, la cosimulació no és massa adequada per a sistemes WLAN, ja que les simulacions, normalment, solen ser molt llargues.

3.5.4. Quantificació de les variables en coma fixa

La quantificació, en el processament del senyal digital, és el procés d'aproximació d'un senyal continu a un conjunt de símbols o valors discrets, definits per la longitud finita de la paraula d'un senyal. L'eina System Generator permet definir el tipus de quantificació (truncar, arrodonir) i de desbordament (saturar, *wrap-around*[1]) quan es defineix la precisió d'un senyal de sortida en els blocs d'entrada i d'operacions (suma, resta, multiplicació,...). Cal tenir en compte la precisió per minimitzar els errors de quantificació i de desbordament. Els errors de quantificació succeeixen quan el número de bits fraccionaris són insuficients per representar la part fraccionària del valor i, els errors de desbordament quan el valor cau fora del rang establert.

A continuació s'expliquen varies tècniques que es poden utilitzar per la determinació del número de bits dels senyals.

- **Comparació del model dissenyat en coma fixa amb el model de Simulink en coma flotant:** aquest sistema s'utilitza sovint tot i que té l'inconvenient que les simulacions són molt llargues. La quantificació es sol fer per blocs i seguint uns criteris basats en mètriques. Un possible criteri seria fitxar un valor del BER (*Bit Error Rate*) i imposar un llindar mínim a la diferencia entre la relació de senyal soroll per al "SNR quantificat" i el "SNR flotant". Llavors, es pot ajustar el número de bits del senyal que es vulgui quantificar a través d'un procés iteratiu que vagi comparant aquesta diferència fins que sigui més petita al llindar establert.

- **Monitorització:** la monitorització del marge dinàmic i el número de bits de les variables és de vital importància en el desenvolupament de sistemes de processament en coma fixa [39]. Aquest procés es pot dur a terme de forma senzilla i eficient a partir de traces temporals en format logarítmic de base 2 i histogrames de variables. El primer cas, essent *x[n]* la seqüència temporal d'una variable i *Vmax* el seu marge de variació, la magnitud de representació en dB2 es defineix com en l'equació (5).

$$x_{dB2} = \log_2 \left| \frac{x[n]}{V_{\max}} \right| \tag{5}$$

D'aquesta manera, la representació temporal de $X_{dB2}[n]$, proporciona de forma visual immediata la informació d'activitat de cadascun dels bits de pes 2^r. El valor de saturació de la representació en complement a dos equival a 0 dB2's. La resta de valors prenen valors negatius respecte aquesta referència. A partir d'aquestes traces es pot extreure el nombre de bits requerits per cada variable (Figura 3.12).

[1] L'efecte *wrap-around* es produeix quan una determinada variable efectua un gir des de la regió més positiva a la regió més negativa de l'interval de representació.

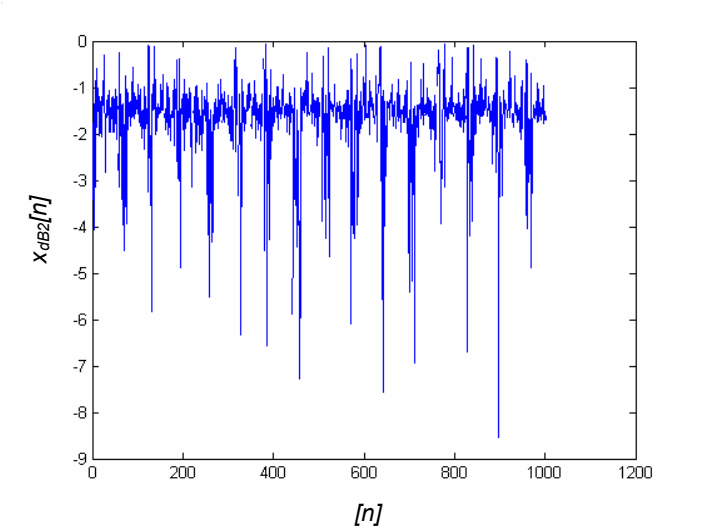


Figura 3.12 Exemple de monitorització del marge dinàmic i de la granularitat d'una variable

Una representació alternativa són els histogrames. Proporcionen idèntica informació que les traces temporals en dB2, però a nivell de probabilitat d'activació de cada bit de la paraula.

3.5.5. Abstracció funcional, ortogonalitat i heterogeneïtat de la metodologia

El System Generator incorpora blocs amb diferents nivells d'abstracció funcional. Aquests poden anar des d'un simple registre a blocs més complexes tals com els blocs IFFT, Viterbi, processadors, ... Aquesta diferència de nivell d'abstracció funcional va força lligada en la utilització de *cores* (optimitzats per a una determinada tecnologia) a l'hora de generar el codi HDL. Això comporta que si es vol un codi font que sigui més transportable tecnològicament, el blocs que s'utilitzin haurien de tenir un nivell d'abstracció funcional baix, tals com registres, retards, ..., ja que el codi HDL que es genera és al nivell de RTL (*Register Transfer Level*) i per tant, transportable.

Un altre aspecte important en les metodologies de disseny al nivell de sistema és el concepte d'ortogonalitat [40]. Aquest concepte s'utilitza per separar diversos aspectes del disseny que permet l'exploració més efectiva de varies solucions. Els casos d'ortogonalitat més utilitzats són entre: funcionalitat i *timing*, funcionalitat i arquitectura i comunicació i computació. Un avantatge importat de l'ortogonalitat és que facilita la reutilització del codi, ja que aquest separa la funcionalitat de la tecnologia. En el cas de la metodologia presentada, l'ortogonalitat és baixa i la més evident és entre la funcionalitat i el *timing*. En el cas d'ortogonalitat entre funcionalitat i arquitectura, cal considerar que aquesta pràcticament no existeix, ja que la majoria de blocs que incorpora System Generator la funcionalitat està molt lligada amb l'arquitectura. De totes formes, si que existeix una ortogonalitat molt evident entre funcionalitat i implementació que permet fer simulacions de la funcionalitat independents de la implementació.

Respecte a l'heterogeneïtat, cal recalcar que la metodologia presentada permet treballar al mateix nivell amb blocs heterogenis, que poden està formats per blocs funcionals de Simulink i de System Generator. També s'hi poden incorporar blocs HDL a través dels blocs *black boxes*[1]. Una altra possibilitat és treballar amb blocs que contenen fitxers de Matlab (*.m). Aquesta heterogeneïtat ens permet treballar al nivell

[1] El *Black Box* és un bloc de la llibreria de Xilinx que permet incorporar mòduls descrits en HDL (VHDL o Verilog) en el model de Simulink/System Generator.

de sistema i amb la possibilitat de fer verificacions heterogènies multinivell (cosimulació, coverificació).

3.5.6. Abstracció del rellotge

El System Generator quan compila un model a hardware preserva la informació dels diferents períodes de mostreig del disseny. Cal remarcar que System Generator utilitza un sol rellotge de sistema síncron per obtenir els períodes múltiples de mostreig. Aquest incorpora diferents senyals d'habilitació de rellotge que controlen varis blocs (*xclock_driver*) que creen els períodes de mostreig que es necessiten. El període de cada un dels senyals d'habilitació dels rellotges és un múltiple enter del període del rellotge del sistema.

Això significa que System Generator ha d'establir una relació entre els períodes de mostreig del model i els períodes corresponents d'habilitació de rellotge que controlen el hardware. Aquesta relació s'estableix a partir dels valors del període de sistema de Simulink i del període del rellotge de la FPGA, especificats pel dissenyador en el bloc System Generator. Aquests valors permeten definir al dissenyador un factor d'escala entre el temps d'una simulació de Simulink i el temps d'una implementació hardware. Cal tenir en compte que un període de sistema de Simulink correcte, és el mínim comú múltiple dels períodes de mostreig que apareixen en el model.

Quan es fa la síntesi d'un sistema amb diversos períodes de mostreig, el System Generator crea un component amb jerarquia superior anomenat *clock_driver*. Aquest fa la crida els diferents components anomenats *xclock_driver* que creen els diferents períodes de mostreig. Llavors, els components *xclock_driver* reprodueixen els períodes de mostreig a partir de registres i lògica combinacional. La següent figura mostra un exemple dels components generats per a un sistema amb dos períodes de mostreig a partir del rellotge de sistema *SYSCLK*.

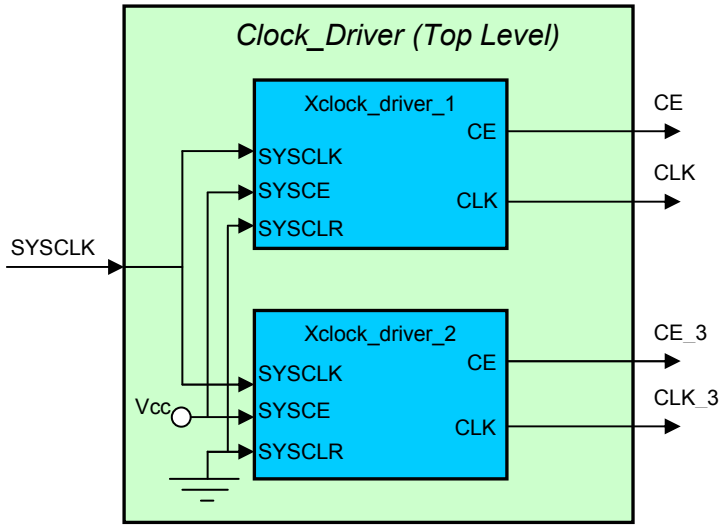


Figura 3.13 System Generator: gestió de rellotges síncrons

3.5.7. Evolució de les eines

Durant el desenvolupament d'aquesta tesi els diferents entorns utilitzats han evolucionat de forma notable. Els entorns relacionats amb les FPGAs que s'han utilitzat són els que més s'han notat (System Generator, ISE, Chipscope). En concret l'eina que ha marcat aquesta evolució ha estat el System Generator, encara que les diferents actualitzacions de l'eina de síntesi ISE afecten també les actualitzacions del

System Generator. S'ha treballat amb l'eina System Generator des del moment que va sortir al mercat, i aquesta, a part de les actualitzacions, ha passat per tres etapes.

La primera etapa va ser la presentació de l'eina on bàsicament era una *toolbox* més de Matlab/Simulink, però amb la capacitat de dissenyar sistemes electrònics per a FPGAs de Xilinx. Aquesta primera versió estava molt centrada en demostrar la funció dels diferents blocs i la seva potència. Un aspecte a destacar d'aquesta versió era que tots els blocs incorporaven una entrada i una sortida de sincronisme de dades entre blocs.

La segona etapa, els de Xilinx, varen eliminar en tots el blocs més elementals la sincronització de dades entre blocs, ja que incrementava considerablement el hardware del disseny. Es van automatitzar l'entrada de blocs HDL, i també, van incorporar més blocs i la possibilitat de fer cosimulacions.

La tercera etapa han integrat altres eines com el Chiscope, la possibilitat de descriure màquines d'estat mitjançant el llenguatge de Matlab (*.m), la gestió de diferents dominis de rellotge (de forma bastant manual), s'han incorporat nous blocs com el picoblaze i s'ha millorat el procés de cosimulació.

Esperem que l'evolució porti a resoldre un dels problemes més grans que presenta aquest entorn que és la gestió dels rellotges (tot i amagant el propi concepte de rellotge a nivell de sistema), sobretot en la part de treballar en diferents dominis.

3.6. Sumari

En aquest capítol s'ha fet una introducció al prototipatge ràpid, definint i classificant el prototipatge. Llavors, segons aquesta classificació el prototipatge que s'utilitza en aquesta tesi s'adapta als sistemes de prototipatge per a plataformes de desenvolupament amb FPGAs. A partir d'aquí s'ha exposat les necessitats de metodologia, arquitectura i tecnologia que necessita una plataforma de prototipatge per un subsistema HiperLAN/2. Un cop definida aquesta plataforma, s'ha exposat l'estat de l'art dels entorns i plataformes vàlides per a sistemes WLAN amb l'objectiu de determinar-ne una metodologia. Cal destacar que s'ha fet una breu presentació de les arquitectures de les FPGAs Virtex de Xilinx. Després s'ha presentat la metodologia utilitzada a la tesi, que va des de Matlab/Simulink a implementació, destacant els mètodes de sincronització, fluxos de disseny i verificació. Cal destacar d'aquesta metodologia els dos fluxos de treball presentats, un més clàssic i l'altre més per a plataformes WLAN específiques. També és important d'aquesta metodologia el gran ventall de possibilitats de verificació que ofereix, sobretot de verificació multinivell.

S'han introduït uns conceptes de prototipatge i tecnològics i s'ha establert una metodologia òptima pel desenvolupament i prototipatge de la capa física que es presenta en els dos propers capítols.

Capítol 4. **Disseny de la capa física**

L'objectiu principal d'aquest capítol és obtenir un model funcional sintetitzable de la part digital de la capa física de l'emissor i del receptor HiperLAN/2, amb una tassa de transmissió de dades de 12MBits/s. En aquests models es presenten diverses aportacions científiques en termes d'implementació d'algorismes de processament de senyal que han pogut ser validades eficientment mercès a la metodologia desenvolupada.

Aquest capítol presenta el disseny de la capa física i s'ha estructurat en diversos apartats. En primer lloc, hi ha una introducció on s'exposen les condicions de partida. Llavors, s'exposen les característiques de l'emissor i del receptor. A continuació, es presenta el disseny separat en dos apartats dedicats a l'emissor i al receptor, explicant cada un dels blocs que els formen. Per fer una explicació més entenedora, tan en l'emissor com en el receptor, aquesta s'ha dividit en quatre subapartats. Al primer, es fa una breu explicació de la funció del bloc i en els casos en que es realitza una aportació científica, una justificació teòrica de l'algorisme. Al segon, es presenten els diferents blocs que formen el disseny. També s'exposen les tècniques de disseny que s'hagin pogut utilitzar. Al tercer, es presenten els resultats de síntesi i simulació del bloc explorat. Al quart (si es dóna el cas), s'exposen les possibles alternatives de disseny comentant els seus principals avantatges i inconvenients. Després es presenta un apartat de resultats, donant resultats globals dels models obtinguts. També es fa una anàlisi de la complexitat del model obtingut i de la seva portabilitat tecnològica. Per acabar es fa un sumari del capítol.

4.1. Introducció

El disseny complert de la capa física d'un sistema WLAN seguint els estàndards HiperLAN/2 o IEEE 802.11a, és un projecte molt complex i llarg que implica la dedicació plena d'un grup d'enginyers durant un cert temps. Per tant, l'esforç de la present tesi, en aquest apartat, està centrat en desenvolupar mòduls de processament de senyal orientats a la seva implementació, i en validar-los, juntament amb la metodologia presentada, i que s'ha mostrat altament eficient per a l'exploració d'algorismes per blocs específics (en aquest cas els més importants) que formen la cadena de transmissió i recepció d'aquests sistemes WLAN.

Per facilitar la comprovació i exploració dels diferents algorismes que intervenen en la cadena, s'ha modelat un emissor i receptor propi en coma flotant a través de Simulink, partint del model presentat per C.Thorpe a l'any 1998 [41]. Aquest model segueix les especificacions de l'estàndard HiperLAN/2, però no inclou el canal, la sincronització temporal i l'estimació de canal.

També, s'han modelat i parametritzat els cinc canals possibles d'HiperLAN/2 (Taula 2.2) [64] utilitzant el bloc de Simulink *Multipath Fading*, conjuntament amb el seu corresponent soroll blanc gaussià i l'efecte *Doppler*. Aquest model de canal, a part de les simulacions globals de l'emissor-receptor, ha permès fer les diferents comprovacions de sincronisme temporal, estimació d'offset freqüencial i d'estimador de canal. En termes metodològics representa igualment la capacitat de dissenyar en un entorn heterogeni en tant que permet coverificar blocs matemàtics amb blocs hardware, descrits a diferents nivells d'abstracció.

4.2. Característiques de l'emissor i receptor

En aquest apartat es presenten les principals característiques de l'emissor sobre el qual s'ha treballat. S'ha comentat en el capítol 2 que l'estàndard solament especifica l'emissor, deixant més llibertat d'exploració pels blocs del receptor. De totes maneres, els estàndards especifiquen la funcionalitat de la cadena, però no especifiquen la seva implementació.

Les característiques principals de l'emissor/receptor implementat es mostren a la següent taula:

Velocitat de transmissió	12MBits/s
Modulació	QPSK
Relació de codificació	1/2
Número de bits de dades per símbol OFDM (N_{DBPS})	48
Número de bits codificats per símbol OFDM (N_{CBPS})	96
Número de bits codificats per cada subportadora (N_{BPSC})	2
Número total de subportadores	64
Número de subportadores amb informació	52
Número de pilots	4
Durada del símbol OFDM	4 µs
Durada del prefix cíclic	800ns
Espai entre subportadores	312,5 KHz
Amplada de banda del senyal	16,66MHz
Freqüència de mostreig	20MHz
Descodificador convolucional de Viterbi	4 bits, 64 estats, decisió soft.

Taula 4.1 Característiques principals de l'emissor/receptor

4.3. Emissor

La Figura 4.1 mostra el model de l'emissor HiperLAN/2 dissenyat. Es pot veure en la cadena d'aquest emissor els diferents blocs que el formen. Aquests blocs són una mica diferents a la cadena de l'emissor mostrada a la Figura 2.8, ja que els blocs s'han agrupat segons les necessitats de disseny i implementació.

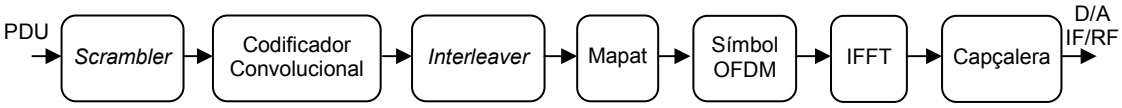


Figura 4.1 Cadena de l'emissor HiperLAN/2

4.3.1. Codificador convolucional

El codificador convolucional és un registre de desplaçament en el que s'introdueixen els bits d'informació a codificar i les sortides són generades com funcions lineals del contingut del registre. Un codificador de raó *1/n*, consisteix en un registre de desplaçament de *M* etapes, amb connexions a sumadors (OR exclusives). Una seqüència d'entrada de L bits produeix una seqüència de sortida de longitud *n*(L+M)* bits. Si un codificador té un registre de desplaçament de M estats, seran necessaris *K=M+1* desplaçaments perquè un bit de missatge entre i surti.

Els estàndards HiperLAN/2 i IEEE 802.11a especifiquen un codificador convolucional amb un *K=7*, una relació de codificació ½ i amb 64 estats. Els polinomis

generadors per a les dues sortides *X* i *Y* són: $G_1=133_{oct}$ per a la sortida X i $G_2=171_{oct}$ per a la sortida Y. L'esquema d'aquest codificador es pot veure a la Figura 4.2.

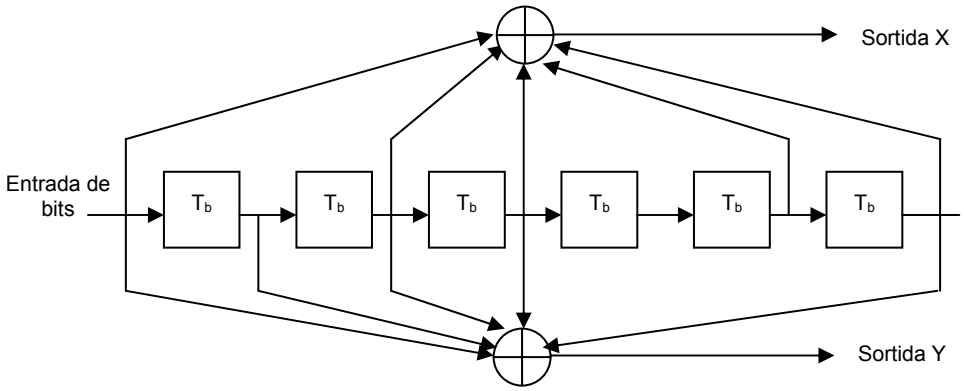


Figura 4.2 Codificador Convolucional de relació de codificació ½

a) Disseny

La llibreria de Xilinx disposa d'un bloc codificador convolucional parametritzable que permet la implementació d'aquest bloc segons els estàndards. Els paràmetres que s'han d'entrar són la *K* i els dos polinomis G_1 i G_2 en format octal. A la Figura 4.3 es mostra el procés de codificació. El tipus de sortida és de dos bits en paral·lel *X1* i *Y1* que s'han de passar a sèrie per enviar-los al bloc *interleaver* amb la sortida *X* en primer lloc. També, disposa d'un bit de sortida que informa quan els bits de la sortida són vàlids. Respecte a les entrades, disposa de l'entrada de bits i una entrada binària que informa el bloc quan el bit és correcte. Existeix la possibilitat d'activar una entrada d'*enable* i una altra de *reset* per quan es dissenyi en el futur la interfície amb la capa MAC.

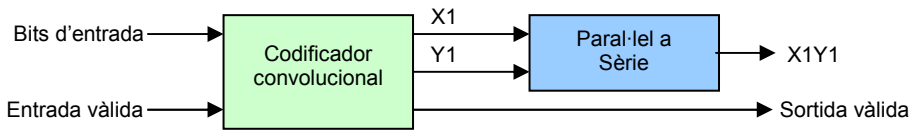


Figura 4.3 Procés de Codificació

b) Resultats

S'ha comprovat que les sortides generades pel bloc convolucional de Xilinx compleixen les especificacions de l'estàndard. El retard de bloc és de 2 cicles de rellotge, ja que enregistra l'entrada i les dues sortides. Per tant, necessita onze registres en total més les taules de memòria (LUT) per fer les funcions or-exclusiva.

c) Alternatives de disseny

L'alternativa senzilla de fer és implementar directament l'esquema de la Figura 4.2, ja que només es necessiten sis registres pel registre de desplaçament, dos registres per enregistrar les dues sortides, un registre per l'entrada i cinc portes or-exclusiva més dos registres per informar que la dada de la sortida és correcta. S'ha fet una comparació dels recursos utilitzats entre el codificador de Xilinx (IP *hard*) i aquesta alternativa i els resultats són iguals.

4.3.2. Interleaver

Els canals WLAN típicament introdueixen els errors per ràfegues i els codis convolucionals, que són els que contempla l'estàndard HiperLAN/2, treballen molt bé per a una distribució d'errors uniforme a les ràfegues. Amb el bloc d'*interleaver* s'aconsegueix una distribució d'errors uniforme. A l'emissor els bits són permutats de forma tal que assegura que els bits adjacents siguin separats diversos bits mitjançant el bloc d'*interleaver*.

Els bits codificats d'un símbol OFDM són permutats en dos passos. El primer pas (*Interleaver* P1), constitueix en una reordenació dels bits entre les subportadores i, el segon (*Interleaver* P2), en una reordenació dels bits que pertanyen a cadascuna de les subportadores.

Si s'anomenen *k* i *i* els índexs d'ordenació d'entrada i de sortida de la primera permutació respectivament, aquesta es pot expressar amb la següent equació:

$$i = (N_{CBPS}/16)(k \bmod 16) + \lfloor k/16 \rfloor, \qquad k = 0,1,\ldots,N_{CBPS}-1 \tag{6}$$

On N_{CBPS} és el número de bits codificats en un símbol OFDM, el símbol $\lfloor\ \rfloor$ indica l'arrodoniment per baix de la part entera i *mod* és el residu de la divisió.

Es pot comprovar l'algorisme utilitzant el programa Matlab donant valors als paràmetres de l'equació , pel cas particular de $N_{CBPS} = 192$ i $k = [0:191]$, l'expressió de l'equació es pot escriure per a Matlab com:

$$i = \left(\left(N_{CBPS}/16\right) * \mathrm{mod}(k,16) + floor\left(k/16\right)\right)' \tag{7}$$

On N_{BPSC} és el número de bits codificats per a cadascuna de les subportadores. El resultat que s'obté és una matriu columna de 192x1 permutada en grups de 16 i de $3*N_{BPSC}$. Aquest bloc d'*interleaver* es pot representar com una matriu rectangular de 16 files i $3* N_{BPSC}$ columnes (Figura 4.4).

$$\begin{array}{c} \\ 16 \end{array} \overset{3*\,N_{BPSC}}{\longleftrightarrow} \begin{bmatrix} 0 & 1 & \cdots & 3*N_{BPSC}-2 & 3*N_{BPSC}-1 \\ 3*N_{BPSC} & 3*N_{BPSC}+1 & \cdots & 6*N_{BPSC}-2 & 6*N_{BPSC}-1 \\ \vdots & \vdots & & \vdots & \vdots \\ 45*N_{BPSC} & 45*N_{BPSC}+1 & \cdots & 48*N_{BPSC}-2 & 48*N_{BPSC}-1 \end{bmatrix}$$

Figura 4.4 Representació Matricial de la permutació P1

Pel cas de la segona permutació, si s'anomenen *i* i *j* els índexs d'ordenació d'entrada i de sortida de la segona permutació respectivament. Llavors, la segona permutació es pot expressar amb la següent equació:

$$j = s*\lfloor i/s \rfloor + \left(i + N_{CBPS} - \left(\lfloor 16*i/N_{CBPS} \rfloor\right) \bmod 2\right) \bmod s, \qquad i = 0,1,\ldots,N_{CBPS}-1 \tag{8}$$

On el símbol $\lfloor\ \rfloor$ indica l'arrodoniment per baix de la part entera, *mod* és el residu de la divisió i *s* es defineix com:

$$s = \max(N_{BPSC}/2,1) \tag{9}$$

Es pot validar l'algorisme en Matlab, donant valors als paràmetres de l'equació (2), pel cas particular de $N_{BPSC} = 4$, $N_{CBPS} = 192$, $i = [0:191]$ i $s = 2$, l'expressió de l'equació (2) es pot escriure per a Matlab com:

$$j = \left(s * floor\left(i/s\right) + \mathrm{mod}\left(i + \mathrm{N_{CBPS}} - \mathrm{mod}\left(floor\left(16 * i/\mathrm{N_{CBPS}}\right),2\right),s\right)\right)' \tag{10}$$

En les modulacions BPSK o QPSK aquesta segona permutació no afecta ja que el número de bits per subportadora (N_{BPSC}) val 1 o 2. En canvi, per a valors més alts de

N_{BPSC}, la trama de bits s'agrupa en blocs de N_{CBPS}/16 bits i els grups senars (1er, 3er, ...) es deixen iguals, mentre que els grups parells (2n,4t, ...) són permutats amb una simple rotació tal com mostra la següent equació:

$$\begin{aligned} 0 &\rightarrow 1 \\ 1 &\rightarrow 2 \\ &\vdots \\ N_{BPSC}/2-1 &\rightarrow 0 \end{aligned} \qquad (11)$$

a) Disseny

Aquest bloc forma part d'un flux de dades continu. Per tant, s'utilitza una memòria RAM de doble port amb paginació (treballant de forma entrellaçada entre dues pàgines: pàgina 0 i pàgina 1). D'aquesta forma mentre s'escriu en una pàgina es pot llegir l'altra simultàniament. La Figura 4.5 mostra el procés d'escriptura i lectura de l'*interleave*r P1 evitant així l'ús de dues memòries (aquest procés està optimitzat per a les tecnologies de les FPGAs).

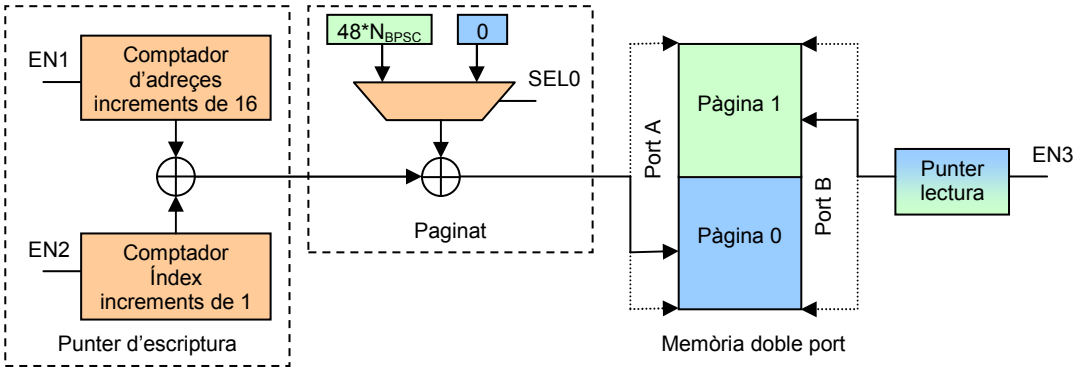


Figura 4.5 Procés d'escriptura i lectura de l'*interleaver* P1

Les dades entren a la memòria a través del port A en salts de 16 posicions controlats pel comptador d'adreces i el comptador d'índex, per tal de passar les dades entrades per files a columnes. D'aquesta forma les dades queden ordenades en grups de 16x1 a punt de ser llegides de forma seqüencial. Les dades es treuen pel port B (ordenades per columnes) controlades pel punter de lectura.

Per tal d'optimitzar els recursos utilitzats de la FPGA, s'ha parametritzat el número de bits dels comptadors que depèn del número d'adreces de les memòries. Aquestes adreces són potència de 2. Llavors, aquest paràmetre es pot expressar per a cada comptador com:

Comptador d'adreces = $ceil(log2(16*3*N_{BPSC}))$

Comptador d'índex = $ceil(log2(16*3*2*N_{BPSC}))$

on *ceil* és la funció de Matlab d'arrodoniment per dalt al número enter de la part decimal.

La unitat de control del procés de lectura i escriptura mostrat a la Figura 4.5 és molt simple. Bàsicament està formada per comparadors, lògica i registres que controlen els *enables* (*EN2, EN3*) del comptador d'índex i del punter de lectura i del senyal de control del multiplexor (*SEL0*). El comptador d'adreces, el controla directament el bit de dada vàlida que arriba del bloc anterior. També, s'afegeix un bit de sortida per validar les dades que són lliurades al pròxim bloc.

b) Resultats

Els mòdul d'*interleaver* obtingut és parametritzable en funció del número de bits per subportadora, facilitant la seva futura integració a l'emissor d'OFDM en funció del mode seleccionat.

El retard del bloc en cicles de rellotge serà el número de bits codificats per símbol OFDM (N_{CBPS}) + 1 *((N_{BPSC}*16*3)+1)*, tal com mostra la següent taula.

N_{CBPS}	Retard (Cicles de Rellotge)
48	49
96	97
192	193
288	289

Taula 4.2 Retard de l'*interleaver* en funció dels bits codificats per símbol OFDM

La Taula 4.3 mostra els recursos del bloc *interleaver* en funció dels bits codificats per símbol OFDM, utilitzant una FPGA Virtex 300. Cal comentar d'aquests resultats que el número de blocs de memòria que es necessiten pels diferents modes de treball, sempre és el mateix. També és important comentar que aquest bloc pot treballar amb bits o amb símbols utilitzant els mateixos recursos, perquè el control no canvia.

N_{CBPS}	48	96	192	288
SLICEs	24	28	30	35
BRAMs	1	1	1	1

Taula 4.3 Recursos de FPGA del bloc *interleaver*

c) Alternatives de disseny

Una alternativa al disseny és utilitzar memòries RAM d'un sol port en lloc de les memòries RAM de doble port. Es complica el procés perquè les adreces s'han de multiplexar per a l'escriptura i la lectura. En canvi, l'avantatge principal d'aquesta solució és que permet utilitzar memòria distribuïda o blocs de memòria dedicada.

Una altra possibilitat és utilitzar un algorisme diferent per la permutació, utilitzant l'*interleaver* convolucional [2], tal com mostra la Figura 4.6. Aquest *interleaver*, escriu cada símbol o bit d'entrada en un dels *K* registres de desplaçament que introdueixen un retard de 0 a *K-1* de la durada del símbol. Mentrestant, els registres de desplaçament són llegits cíclicament per fer la permutació dels símbols.

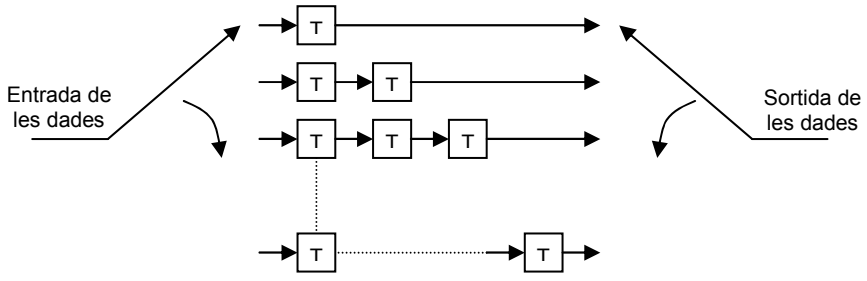


Figura 4.6 *Interleaver* Convolucional

4.3.3. Mapatge

La sortida binària del bloc *interleaver* es divideix en grups de bits de 1, 2, 4 o 6 i convertits en números complexos representats en punts a la constel·lació per BPSK, QPSK, 16QAM o 64QAM. La conversió es fa utilitzant codificació Gray per facilitar el Desmapatge. La Figura 4.7 mostra la codificació en la constel·lació QPSK.

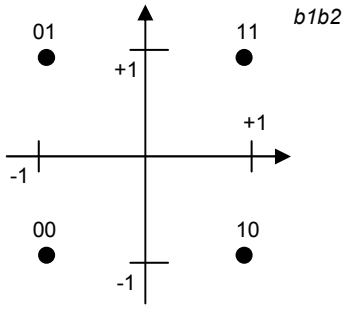


Figura 4.7 Constel·lació QPSK

BPSK			
Bit b1	I		Q
0	-1		0
1	1		0
QPSK			
Bit b1	I	Bit b2	Q
0	-1	0	-1
1	1	1	1

16QAM			
Bit b1b2	I	Bit b3b4	Q
00	-3	00	-3
01	-1	01	-1
11	1	11	1
10	3	10	3
64QAM			
Bit b1b2b3	I	Bit b4b5b6	Q
000	-7	000	-7
001	-5	001	-5
011	-3	011	-3
010	-1	010	-1
110	1	110	1
111	3	111	3
101	5	101	5
100	7	100	7

Taula 4.4 Taules de codificació per BPSK, QPSK, 16QAM i 64QAM

La Taula 4.4 mostra els diferents símbols i la seva codificació en codi Gray pels diferents tipus de modulació. Aquests símbols en format complex (I + jQ) es normalitzen en funció de la modulació que es faci servir amb una factor K_{MOD}, tal com mostra la següent taula:

Modulació	K_{MOD}
BPSK	1
QPSK	$1/\sqrt{2}$
16QAM	$1/\sqrt{10}$
64QAM	$1/\sqrt{42}$

Taula 4.5 Factor de normalització depenent de la modulació

a) Disseny

El component principal d'aquest bloc és una memòria ROM on hi ha guardats, en funció de la modulació, els símbols de la Taula 4.4. Per adreçar aquests símbols, els bits que arriben en sèrie s'organitzen en grups de 1,2,4 o 6 bits a través d'un bloc sèrie-paral·lel, en funció de la modulació. Els primers bits adrecen la memòria ROM *I* i els segons adrecen la memòria *Q*. Els símbols són normalitzats per una factor K_{MOD} mitjançant un bloc multiplicador de Xilinx. Aquest multiplicador ofereix diferents opcions d'implementació que són: números de bits i decimals de la constant, tipus de sortida (complement A2 o sense signe), número de bits i decimals de la sortida, tipus de multiplicador (paral·lel o seqüencial), tipus de memòria (distribuïda o bloc de memòria), paral·lelisme temporal i retard. El tipus de sortida i número de bits de la

sortida depèn de les característiques del pròxim bloc. Al ser parametritzable s'ha optat per deixar una configuració que utilitzi memòria distribuïda, una arquitectura paral·lela espacial i sense paral·lelisme temporal.

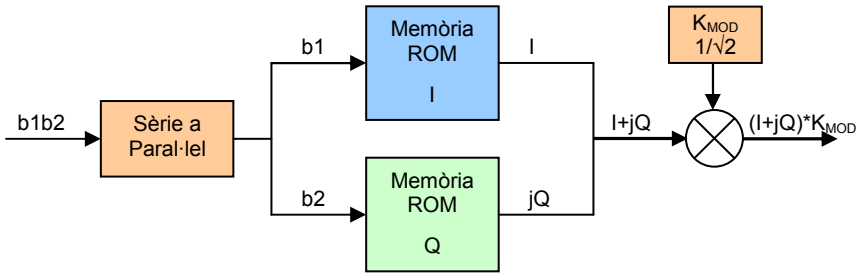


Figura 4.8 Bloc de modulació QPSK

b) Resultats

El bloc de mapat és parametritzable en funció del número de bits per subportadora (N_{BPSC}). El número de bits de la sortida de les memòries ROM és de 4 per poder representar el símbol més gran que és el -7. La taula següent mostra els recursos de FPGA utilitzats del bloc de modulació sense el multiplicador pels diferents casos de modulació.

Modulació	QPSK	16QAM	64QAM
SLICEs	13	15	21
Flip-Flops	15	19	27
LUTs	9	10	15

Taula 4.6 Recursos de FPGA del bloc de Mapat

c) Alternatives de disseny

L'alternativa de disseny és utilitzar multiplexors en lloc de les memòries ROM. El problema que presenta aquesta alternativa és la dificultat de parametrització dels multiplexors. De totes maneres, s'ha explorat el cas de QPSK amb multiplexors i els resultats obtinguts són molt similars: 9 *Slices*, 15 FF (*Flip-Flops*) i 9 LUTs.

4.3.4. Símbol OFDM

El bloc símbol OFDM és l'encarregat d'ordenar les dades per a les subportadores, inserir els pilots, la component de contínua i les subportadores laterals que no porten informació segons les especificacions dels estàndards HiperLAN/2 i IEEE 802.11a. També, sincronitza les dades que arriben del bloc de mapat a 12 Mbits/s amb les que s'envien al bloc IFFT a 60 Mbits/s.

A continuació es presenten de forma resumida aquestes especificacions.

Cal recordar, segons les especificacions mostrades a la Taula 4.1, que un símbol OFDM està format per 64 subportadores. D'aquestes 48 porten informació i 4 són el símbols pilots.

Si l és el número de subportadora, llavors els índexs de les subportadores que porten informació són:

$$-26 \le l \le -22, -20 \le l \le -8, -6 \le l \le -1, 1 \le l \le 6, 8 \le l \le 20, 22 \le l \le 26$$

I els índexs de les subportadores pilot són:

$$l = -21, -7, 7, 21$$

La subportadora de l'equivalent passa baixes que s'escau en continua (posició zero) no s'utilitza i val zero.

Si n és el símbol OFDM i $C_{l,n}$ és el símbol complex (dades o pilots) per a la subportadora l i pel símbol OFDM n, llavors la distribució del grup de dades $D_{m,n}$ a dins dels símbols $C_{l,n}$ es mostra a la següent figura:

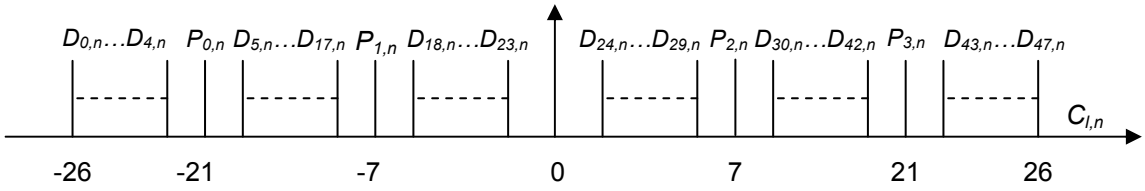


Figura 4.9 Distribució de les dades a les subportadores

Per aconseguir aquesta distribució, cal ordenar les dades que arriben al bloc IFFT tal com mostra la Figura 4.10. Es pot veure que la primera dada és el zero corresponent a la posició de continua de l'espectre de l'equivalent passa baixes. Cal comentar que les 48 dades ($D_{0,n}...D_{47,n}$) que arriben per símbol OFDM són reordenades. La reordenació consisteix en agafar les 24 dades últimes i passar-les al principi. També, es pot veure que s'insereixen zeros entre les posicions 26 i 38 que corresponen a les bandes de guarda laterals de l'espectre.

La Figura 4.9 mostra les posicions dels quatre símbols pilots ($P_{l,n}$) que s'insereixen en les subportadores corresponents. El senyal de referència transmès es defineix com:

$$C_{l,n} = \begin{cases} +p_n, l = -21 \\ +p_n, l = -7 \\ +p_n, l = 7 \\ -p_n, l = 21 \end{cases}$$

on p_n és una seqüència pseudo aleatòria amb una extensió de 127 elements donats per la següent seqüència:

$$p_{0...126} = \begin{Bmatrix} 1,1,1,1,-1,-1,-1,1,1,-1,-1,-1,1,1,1,-1,1,1,-1,-1,1,1,1,-1,1,1,-1,1,1,-1,-1,-1,1,1,-1,1,1,-1,1,1,1,1,1,1,1,-1,1,1,1,-1,1,1,-1,-1,1,1,1,-1,1,-1, \\ -1,-1,1,1,-1,1,1,-1,-1,1,1,-1,-1,-1,1,1,1,1,1,1,1,-1,-1,1,1,1,-1,-1,1,-1,1,1,-1,1,1,1,-1,-1,-1,1,1,-1,-1,1,1,-1,1,1,-1,-1,-1,1,1,-1,-1,-1,1,1, \\ -1,1,1,1,1,1,-1,1,1,-1,1,1,-1,1,-1,-1,-,1-,1-,1,1,-1,1,1,1,-1,1,1,-1,1,1,-1,1,1,1,-1,-1,1,1,-1,1,1,-1,-1,-,1,1,1,1,1,-1,-1,-1,-1,-1, \\ -1,-1 \end{Bmatrix} \quad (12)$$

Aquesta seqüència es generada a partir del següent polinomi:

$$S(x) = X^7 + X^4 + 1 \quad (13)$$

L'estat inicial del generador han de ser tots "1" i s'ha d'inicialitzar al principi de totes les unitats de dades (PDUs). Llavors, abans d'enviar els pilots al bloc IFFT, tots els valors "0" i "1" són substituïts per valors "1" i "-1" respectivament. Per a cada símbol OFDM s'utilitza un element de la seqüència.

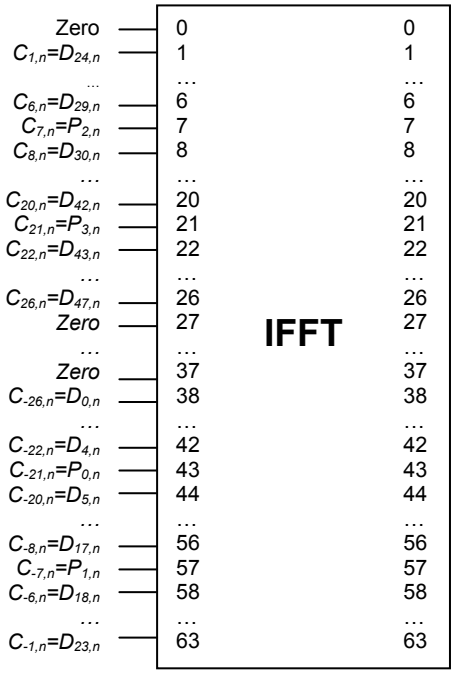


Figura 4.10 Ordenació de les dades pel bloc IFFT

a) Disseny

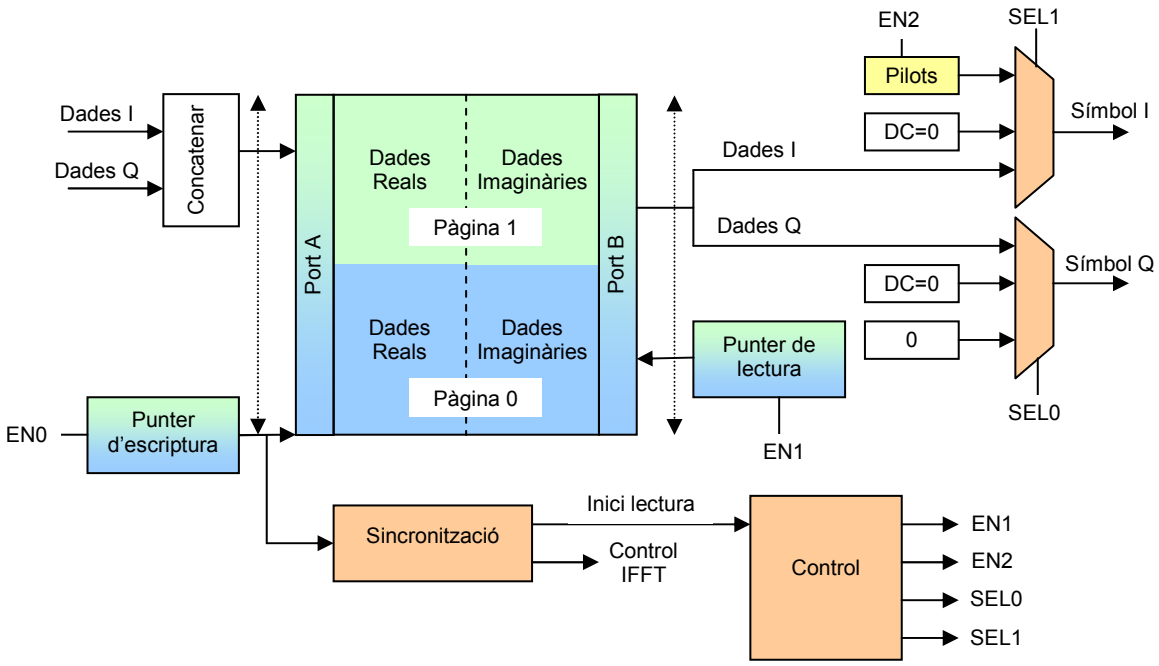


Figura 4.11 Arquitectura del bloc Símbol OFDM

La Figura 4.11 mostra l'arquitectura del bloc Símbol OFDM que està formada per una unitat de procés i una unitat de control.

La unitat de procés li arriben cíclicament 48 valors complexos per a cada símbol OFDM. Llavors, la part real i la imaginària d'aquests símbols són concatenats i guardats a la mateixa posició de la memòria per estalviar recursos. Per mantenir un

flux continu de dades, s'utilitza una memòria RAM de doble port d'una profunditat de 128 posicions amb paginació. Per ordenar les dades segons la Figura 4.10, primer s'insereix la component de continua, en aquest cas '0', mitjançant els multiplexors de sortida. A continuació es llegeixen les dades de la pàgina de memòria corresponent a partir de la posició 23 a través del punter de lectura, per tal d'intercanviar el grup de les primeres 24 dades pel grup de les últimes 24 dades. Durant aquest procés de lectura s'insereixen els pilots a les subportadores corresponents, inhibint el punter de lectura i controlant els multiplexors de sortida. Per inserir els zeros a les subportadores laterals (banda de guarda), es llegeixen les posicions restants, inicialitzades amb zeros, entre les 64 posicions d'una pàgina de memòria i les 48 dades gravades. La següent figura mostra l'ordre que queden les dades abans d'entrar al bloc IFFT.

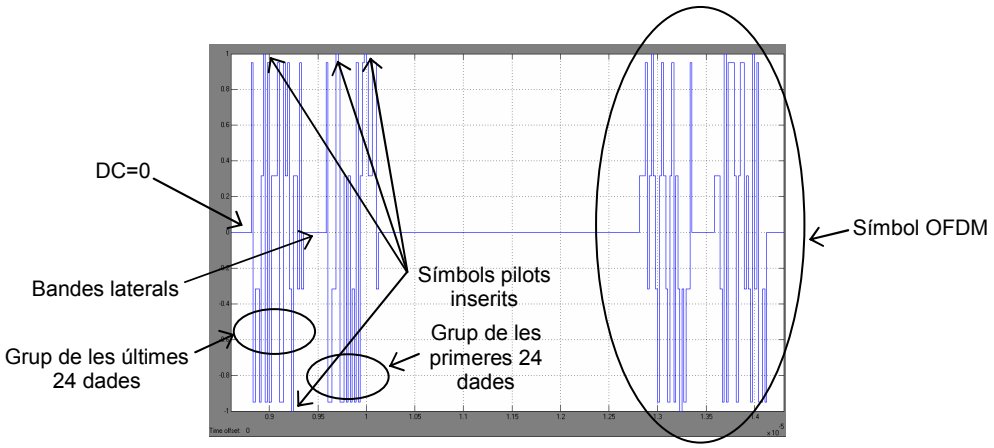


Figura 4.12 Distribució de les dades a les subportadores

El punter d'escriptura està format per un comptador de mòdul 48 amb un registre i lògica combinacional per gestionar l'entrellaçat. El punter de lectura està format per un comptador de mòdul 64 amb un valor d'inici de 24 i amb entrades de recàrrega i *reset*. També, inclou tres registres i lògica combinacional per controlar el paginat, la recàrrega i el *reset*.

El bloc pilot, és el bloc encarregat de generar la seqüència pseudoaleatòria de l'equació (13). Aquesta seqüència es pot generar a partir de registres i portes or-exclusives, tal com mostra la següent figura:

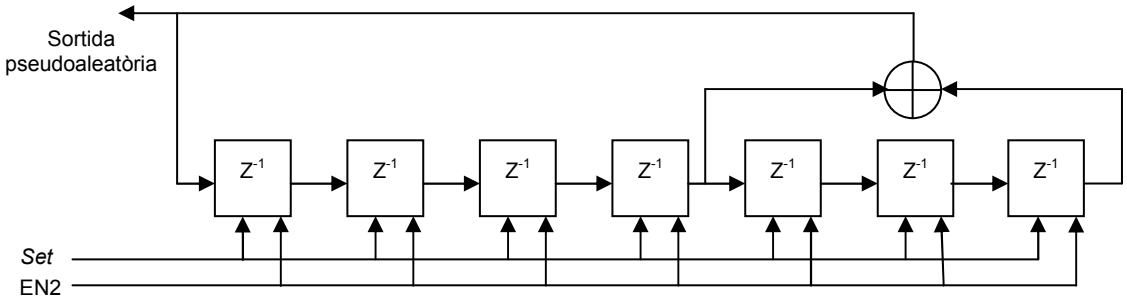


Figura 4.13 Generador pseudoaleatòri p_n

Per aconseguir l'estat inicial de tots uns, els registres disposen d'una entrada de *set*. També, s'inclou una habilitació (*enable*) per controlar la generació de la seqüència un cop per símbol OFDM. Llavors, només s'ha de mapar la seqüència de sortida afegint un multiplexor a la sortida per obtenir els '1' i '-1' de la seqüència (12) (Figura 4.14).

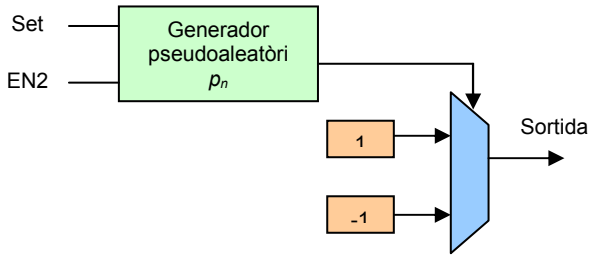


Figura 4.14 Generador pseudoaleatòri amb mapat

La unitat de control està formada per dos blocs: el bloc de sincronització i el bloc de control.

El bloc de sincronització, sincronitza temporalment el principi de la trama de les dades de la sortida, en funció de les dades d'entrada (*EN0*) (Figura 4.15). Aquest bloc dóna l'ordre de començar al bloc de control quan la pàgina 0 de memòria s'ha acabat d'escriure per primer cop. Llavors, comença el procés de lectura de la pàgina 0 i continua el procés d'escriptura a la pàgina 1. Cal tenir en compte que la velocitat d'entrada i de sortida de les dades és diferent, per tant, el bloc de sincronització ha de reduir el valor del període de mostreig per un factor equivalent a la relació entre les dues velocitats.

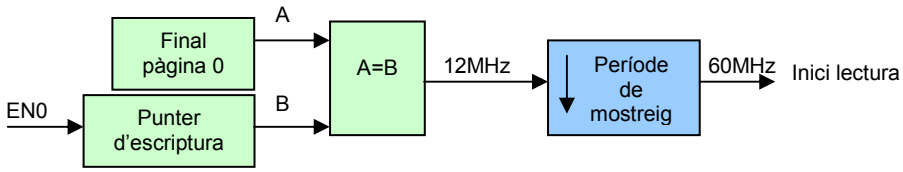


Figura 4.15 Procés de sincronització de lectura

Aquest procés està format per un comparador i un component IP (bloc de Xilinx) enregistrat per baixar el període de mostreig.

Simultàniament, al procés de lectura, es genera una sortida de control per al bloc IFFT que l'informa de quan comença cadascun dels símbols OFDM. Aquesta sortida de control és un senyal de 64 cicles de rellotge de durada que s'anirà repetint per a cada símbol OFDM. Si un símbol OFDM dura 4µs i la freqüència de mostreig és de 60MHz, llavors es necessiten 240 cicles de rellotge per obtenir un símbol OFDM. Aquest procés està format per un comptador de mòdul 240, registres i dos comparadors.

El bloc de control genera tots els senyals de control per a la unitat de procés. Aquest bloc està format per un comptador de mòdul 64 i diversos comparadors que van detectant les posicions on cal actuar. Llavors, en funció d'aquestes posicions es generen els diferents senyals de control.

b) Resultats

La següent taula mostra els recursos que utilitza aquest bloc. Cal destacar els 71 *flip-flops* que són deguts majoritàriament als comptadors. El bloc de memòria correspon a la memòria RAM de doble port.

SLICEs	152
Flip-Flops	71
LUTs	258
BRAMs	1

Taula 4.7 Recursos de FPGA del bloc símbol OFDM

A continuació es presenten les simulacions realitzades amb l'entorn Matlab/Simulink i System Generator dels punters de lectura i d'escriptura.

La següent figura mostra la simulació del punter de lectura amb Simulink. Cal destacar (a la part alta de la figura) l'evolució temporal del punter on es pot veure clarament el canvi de pàgina de memòria. A la part baixa de la figura, es mostren en detall les posicions on es produeixen les insercions dels símbols pilots i de la component de continua.

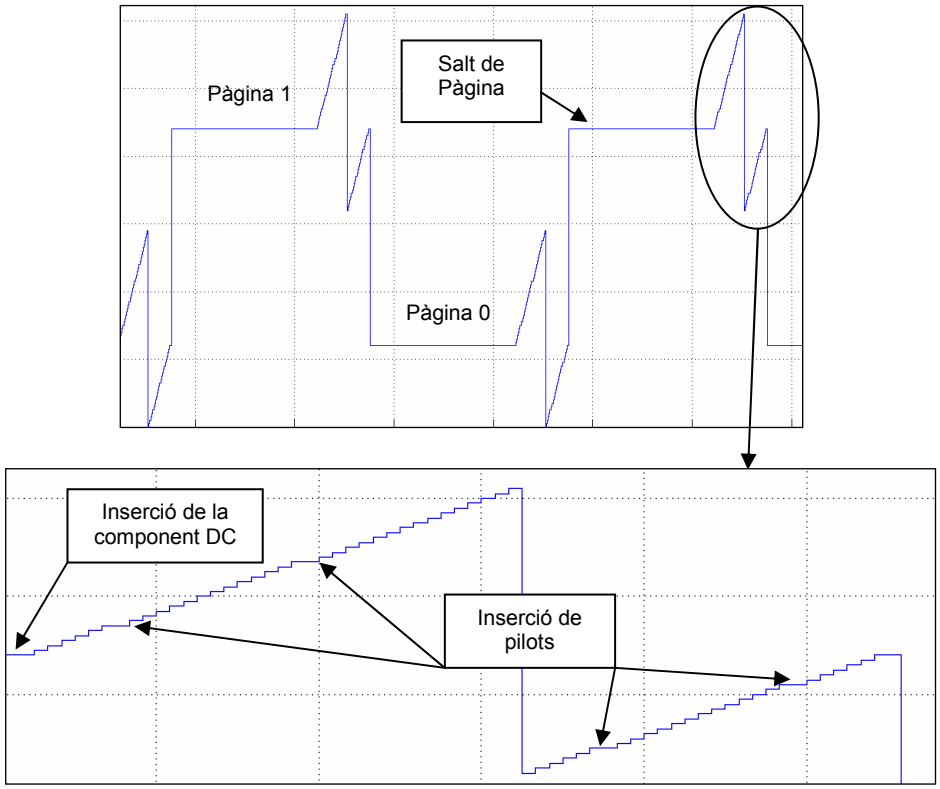


Figura 4.16 Evolució temporal del punter de lectura

La Figura 4.17 mostra la simulació del punter d'escriptura amb Simulink. A la figura es pot veure l'evolució temporal del punter i els salts de pàgina de memòria.

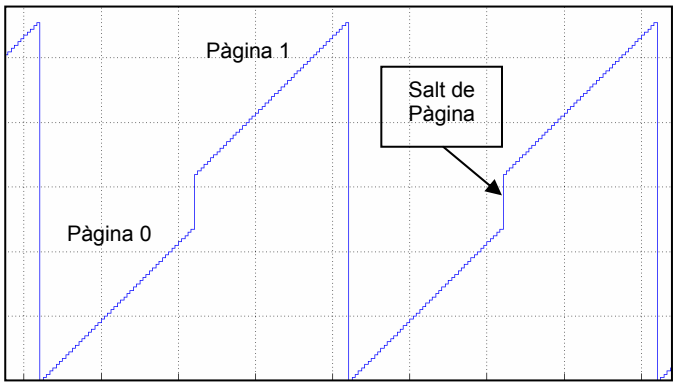


Figura 4.17 Evolució temporal del punter d'escriptura

c) Alternatives de disseny

Les alternatives de disseny poden ser molt diverses, ja que està format per petites parts i cadascuna d'elles pot tenir la seva pròpia alternativa de disseny. A nivell global del bloc, la unitat de control es pot implementar a través d'una màquina d'estats en lloc d'utilitzar comptadors, registres i lògica combinacional. Les llibreries de System Generator disposen de blocs amb màquines d'estats que podrien fer aquesta funció, tot i que no creiem que això reporti guanys substancials.

4.3.5. IFFT/FFT

El blocs IFFT/FFT són els encarregats de modular i desmodular els símbols OFDM complexos en les subportadores. El bloc FFT correspon al receptor però s'ha considerat adient incloure la seva explicació en aquest apartat, ja que la seva estructura és molt semblant al bloc de la IFFT.

En aquest mòdul concret s'ha derivat un algorisme de *radix 2* que presenta propietats interessants per la seva implementació en hardware, tals com el nivell de control del paral·lelisme i una arquitectura idèntica en cadascuna de les etapes amb una mateixa realimentació entre les connexions de les etapes continues. La contribució passa per canviar els gràfics de flux de dades que descriu el clàssic algorisme de Cooley-Tukey [45] i readaptar-lo amb l'objectiu d'obtenir un nou ordre, per trobar l'algorisme utilitzant una notació matricial en la seva representació, partint de les propietats de la transformada discreta de Fourier, conegudes amb el nom de delmació en temps i en freqüència (apèndix A). L'algorisme de la FFT/IFFT és derivat en termes de productes de matrius on cadascuna de les matrius representa una etapa o una matriu de permutació de reordenació. El resultat dóna una representació gràfica del procés de les interconnexions entre etapes. Per aconseguir etapes amb una mateixa arquitectura, s'insereix entre les etapes combinacions de matrius de permutació parell-senar $P(n)$ multiplicada per la seva inversa $P(n)^{-1}$ per anul·lar el seu efecte [47].

Les matrius introduïdes entre les etapes no incrementen la complexitat matemàtica de l'algorisme, ja que aquesta està associada al número de multiplicacions. Les matrius de permutació només aporten una modificació dels pesos de cada etapa que permet canviar la connexió entre elles.

Per exemple, per a una FFT de 8 punts, cada etapa consta de quatre blocs anomenats papallona (*butterfly*) (Figura 4.19) que representen la combinació lineal de dues sortides amb només dues entrades, facilitant d'aquesta forma el processament en paral·lel. El nou gràfic de flux de dades presenten estructures idèntiques, però amb el valor dels seus pesos canviats respecte a l'algorisme clàssic de Cooley-Tukey, tal com mostra la Figura 4.18.

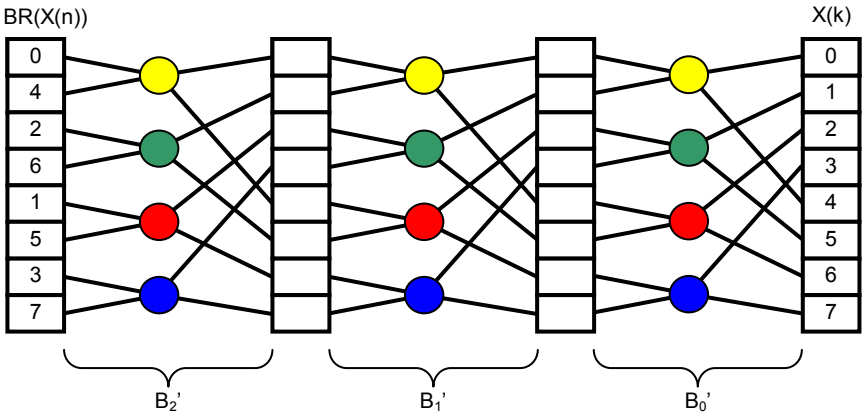


Figura 4.18 Diagrama de flux de l'algorisme implementat per a una FFT de 8 punts

Una papallona bàsica de dues entrades i dues sortides només necessita un multiplicador i dos sumadors complexos. El pes de cada papallona és una constant (*α*) i varia per a cadascuna d'elles. A la Figura 4.19 es mostra l'estructura d'aquesta papallona.

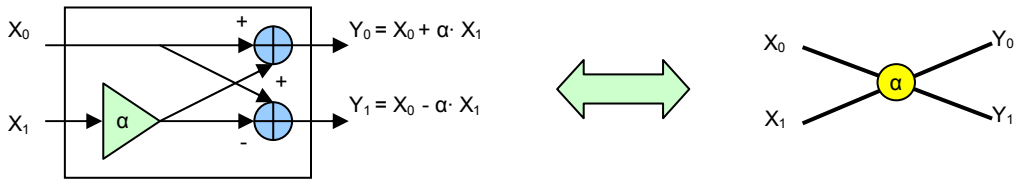


Figura 4.19 Estructura bàsica d'una papallona (*Butterfly*)

a) Disseny

Segons les especificacions dels estàndards HiperLAN/2 i IEEE802.11a, la IFFT/FFT ha de ser de 64 punts. Aquest estàndards no diuen res sobre la longitud de les paraules, número de bits decimals o velocitat de processament. El número de bits de la sortida de la IFFT es pot adaptar al convertidor digital/analògic que es disposi, en el cas de la plataforma Nallatech és de 14bits. Normalment els dissenys d'OFDM, el número de bits oscil·len entre el 10 i 14bits. En aquest cas s'ha decidit fer una IFFT/FFT amb el número de bits parametritzable i amb dos bits enters per cobrir un rang d'entrada màxim de -1 a 1 i evitar possibles saturacions en les operacions aritmètiques de l'algorisme.

Respecte a la velocitat de processament, cal tenir en compte que s'ha de mantenir un flux de dades continu amb una velocitat de dades d'entrada del bloc anterior a 60MHz i de sortida a 20MHz. La velocitat que s'ha escollit és de 60MHz perquè és múltiple de l'entrada i de la sortida i és acceptable per implementar en una FPGA.

L'algorisme presentat permet una gran flexibilitat en termes de paral·lelisme [47][48][49][50], però per implementar tota l'arquitectura en paral·lel es necessiten molts de recursos, per tant, s'ha cercat una solució de comprimís entre els recursos de la FPGA i el temps de processament de l'algorisme. El número d'etapes que es necessiten per completar una IFFT/FFT de 64 punts utilitzant una papallona de dues entrades i dues sortides és de 6 ($log_2(64)$) i el número de papallones per etapa és de 32. De les agrupacions possibles de papallones de 2, 4, 8, 16 o 32 s'ha escollit la de 4. Les opcions de 8, 16 o 32 s'han descartat perquè presenten un excessiu paral·lelisme respecte dels requeriments temporals d'aquesta aplicació, ja que es disposa de temps suficient (3,2μs) per adquirir, processar i treure les dades.

Les 4 papallones fan el processament en paral·lel, per tant, per completar una etapa de 32 papallones, cal fer el procés 8 vegades. Finalment, per obtenir el resultat global de la IFFT/FFT cal repetir aquest procés per a cadascuna de les etapes, és a dir, s'ha de repetir 5 vegades més. La Figura 4.20 mostra el diagrama d'aquest algorisme.

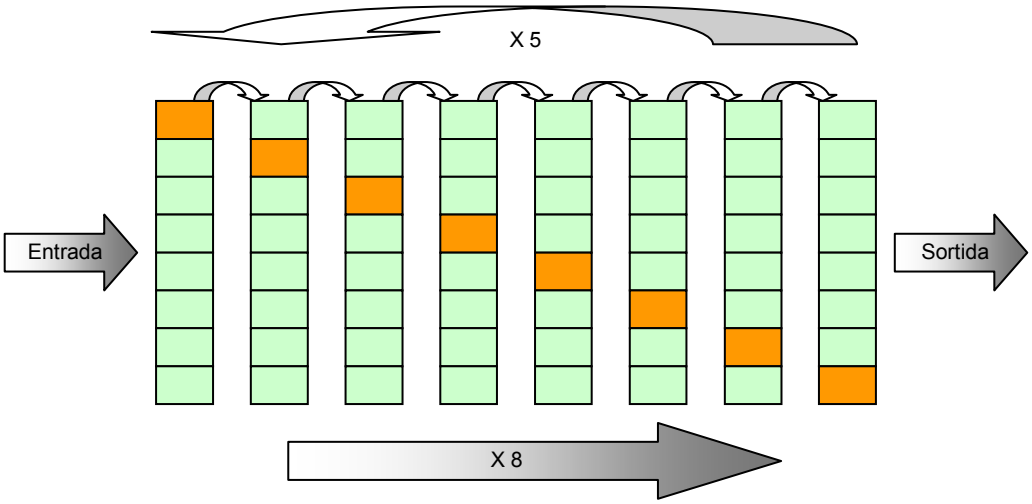


Figura 4.20 Diagrama de l'algorisme de processament per una IFFT/FFT de 64 punts

L'operació de processament consta de tres processos: adquisició, processament i extracció de dades. Només el procés de computació és idèntic entre l'algorisme de la IFFT i FFT, ja que els processos d'adquisició i d'extracció inclouen la inserció i extracció del prefix cíclic que els fan diferents.

Per a la IFFT (Figura 4.21), el senyal de principi de trama s'activa durant 64 cicles de rellotge al principi de cada símbol OFDM. El procés d'adquisició de dades (64Mmostres/s) dura 64 cicles de rellotge (un cicle de rellotge per a cada dada) i es sincronitza amb el senyal de principi de trama. El processament dura 160 cicles de rellotge, 96 cicles són de processament i 64 per treure el resultat. Els 96 cicles estan formats per 48 (8x6) períodes de processament, escombrat vertical i horitzontal, on aquests períodes es desglossen en dos cicles de rellotge, un de lectura i un altre d'escriptura. Aquests dos processos treballen en sèrie amb un rellotge de 60MHz. L'extracció de dades és contínua a una velocitat de sortida de 20MHz i cada símbol OFDM està format per 80 dades (64 corresponen al símbol OFDM i 16 al prefix que s'insereix). Cal recordar que aquest prefix són les 16 últimes dades del símbol OFDM.

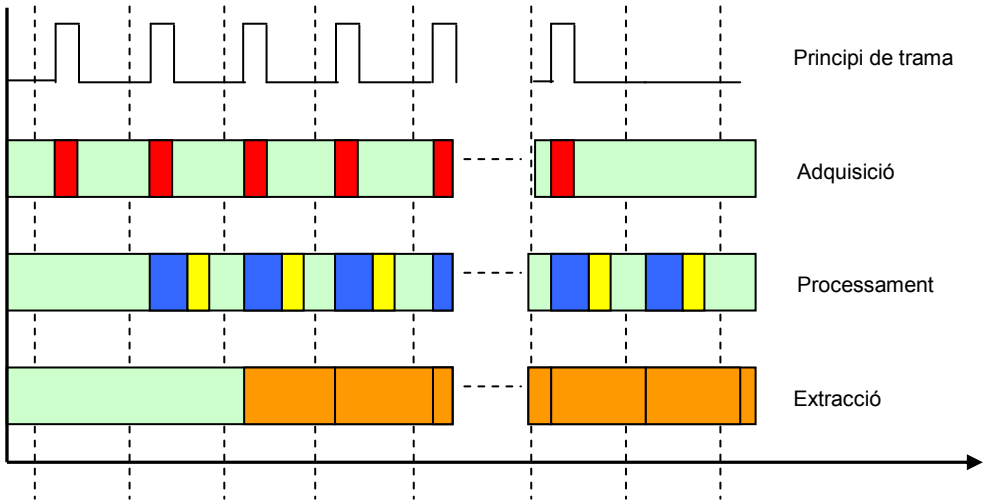


Figura 4.21 Diagrama temporal dels processos d'adquisició, processament i extracció de dades per a la IFFT

Per a la FFT (Figura 4.22), el senyal de dades vàlides es manté actiu durant la durada d'una ràfega. El procés d'adquisició de dades (20Mmostres/s) sincronitzat amb el senyal de dades vàlides dura 80 cicles de rellotge, on les primeres 16 dades són

eliminades perquè corresponen al prefix cíclic inserit en l'emissor. El processament és idèntic a la IFFT i es realitza a 60MHz. L'extracció de dades és contínua a una velocitat de sortida de 16MHz i cadascun dels símbols OFDM desmodulats està format per 64 dades.

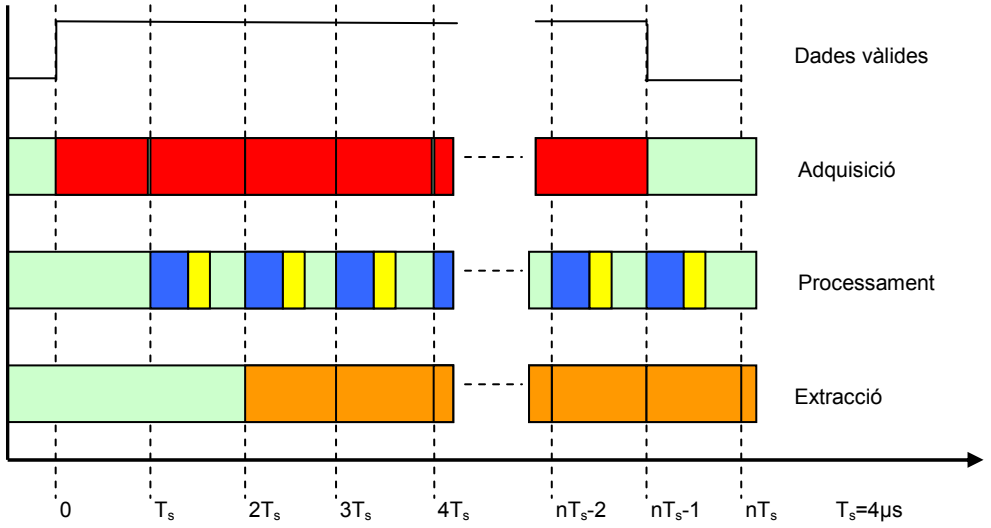


Figura 4.22 Diagrama temporal dels processos d'adquisició, processament i extracció de dades per a la FFT

Adquisició de dades de la IFFT/FFT

La diferència del procés d'adquisició entre la IFFT i FFT, és que la FFT ha d'eliminar les 16 dades del prefix cíclic, però per la resta del procés és idèntic. Al procés d'adquisició, les dades arriben des del bloc símbol OFDM sincronitzades per un bit de començament de trama, on cada trama es correspon a un símbol OFDM amb 64 símbols complexos. Llavors, les parts reals i imaginàries d'aquests símbols són concatenats i guardats en una memòria RAM de doble port. Per mantenir un flux de dades continu, la memòria es pagina en dues pàgines i s'alternen la seva escriptura periòdicament per a cada símbol OFDM, tal com mostra la Figura 4.23.

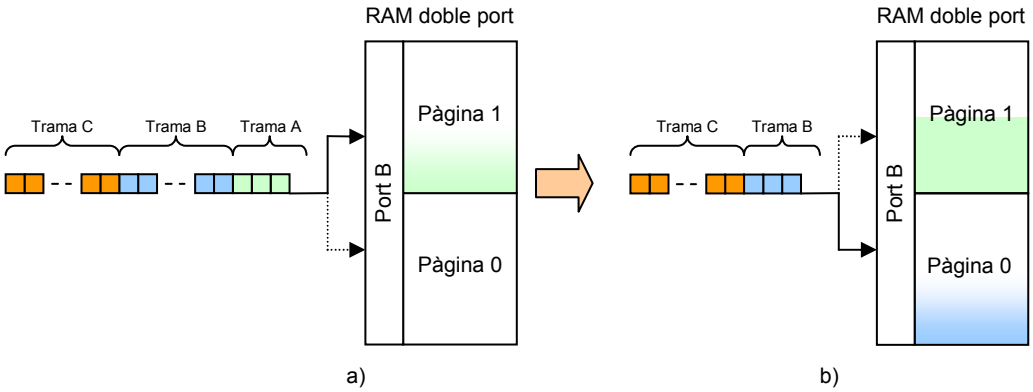


Figura 4.23 Diagrama d'adquisició: paginat del port B. Sigui a) en l'instant T i b) en T+4μs

Processament IFFT/FFT

Aquest procés està format per dos subprocessos en sèrie: el càlcul de la IFFT/FFT i la posterior gravació del resultat a una memòria RAM de doble port diferent de la memòria d'entrada.

Per calcular la IFFT/FFT, no és possible gravar les sortides del processament parcial en el mateix lloc on es trobaven les entrades, ja que es perdria informació que encara no s'ha fet el processament. Per tant, és necessari partir les dues pàgines 0 i 1 en quatre pàgines per tal de fer un moviment de dades: pàgina 1a, pàgina 1b, pàgina 0a i pàgina 0b. Aquest procés es pot veure per a la pàgina 1 a la Figura 4.24. Un cop acabat el procés per a la pàgina 1 es faria el mateix per la pàgina 0. El sistema de paginat permet estalviar blocs de memòria, ja que només s'utilitza un bloc per les quatre pàgines.

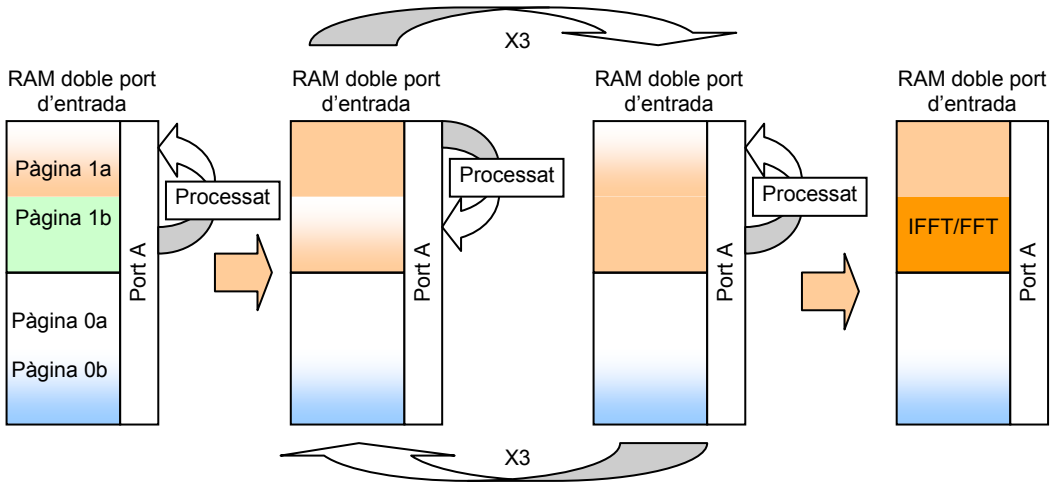


Figura 4.24 Diagrama seqüencial del processament de la IFFT/FFT

Un cop s'ha acabat el processament, el resultat queda guardat a la memòria RAM de doble port de l'entrada. Llavors, són necessaris 64 cicles de rellotge més per enviar aquests resultats a la memòria RAM de doble port de la sortida. Aquesta memòria de sortida també està paginada en dues pàgines per mantenir un flux de dades continu. Aquest procés es pot veure a la figura següent:

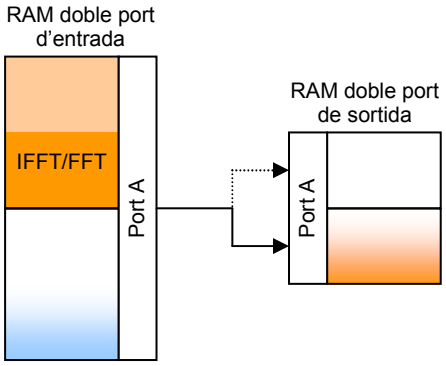


Figura 4.25 Procés de gravació del resultat de la IFFT/FFT a la memòria de sortida

Extracció de dades

Un cop s'ha omplert una pàgina de la memòria de sortida, es comencen a extreure les 80 dades que formen un símbol OFDM pel port B a una velocitat de 20MHz. Cal tenir en compte que hi ha només 64 dades guardades i que les 16 restants són el prefix cíclic que s'insereix. Aquest procés d'inserció es podria fer després, però per estalviar recursos es fa dins del procés de la IFFT (l'extracció de dades del bloc FFT és idèntic excepte el procés d'inserció del prefix cíclic). Aquest procés es pot veure en més detall a la figura següent:

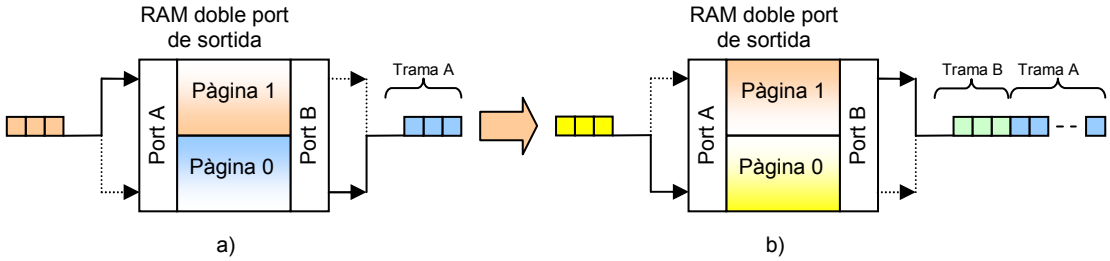


Figura 4.26 Diagrama d'extracció. Sigui a) en l'instant T i b) en T+4μs

Estructura de Hardware

L'estructura de Hardware s'ha dividit en dues unitats: la unitat de control i la unitat de procés.

La unitat de control genera els senyals que regulen el flux de la unitat de procés i està formada per una màquina d'estats finits tipus *Moore*, registres i lògica combinacional. La màquina d'estats finits és molt senzilla i disposa només d'una entrada per controlar el canvi d'estats. Els paràmetres que inclou la màquina d'estats són una matriu que conté els estats propers, el període de treball, un vector de sortida i la seva amplada en bits, on cada bit correspon a una sortida de control. Les sortides de la màquina d'estats controlen comptadors i memòries ROM, les quals generen altres senyals de control per la unitat de processament.

La unitat de procés està subdividida en diversos blocs més simples que són: control de paginat, memòries RAMs de doble port, memòries ROMs, unitat de processament, realimentacions i control de sortida (Figura 4.27).

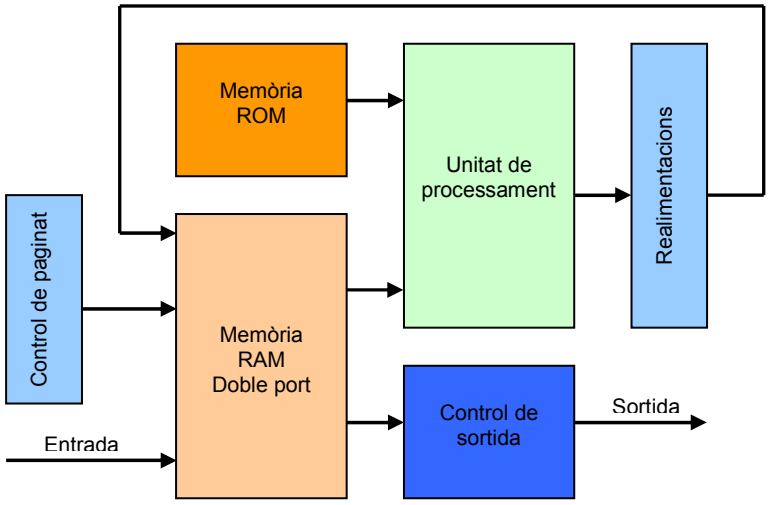


Figura 4.27 Unitat de procés

El bloc de control de paginat controla l'accés a les diferents pàgines de la memòria RAM de doble port d'entrada (veure Figura 4.24) durant els processos d'adquisició i processament. El bloc està format per blocs de concatenació que afegeixen 2 bits a les adreces de la memòria. Els valors d'aquests bits són fixats per la unitat de control.

El bloc de memòria RAM de doble port s'utilitza per guardar les 64 dades de 2·nbits (part real i imaginaria concatenada) de longitud que cal processar durant el temps de símbol OFDM. També, guarda els resultats parcials del procés. El bloc està format per un conjunt de 8 memòries RAM de doble port, perquè les quatre papallones de l'element de procés necessiten 8 dades en paral·lel. La profunditat de les memòries és

de 32 paraules i per tant, la profunditat total de les 8 memòries és de 256 paraules. En realitat, només seria necessària una profunditat de 64, però els diferents paginats la quadruplica. Finalment, l'accés a les diferents pàgines i memòries es controla a través de la unitat de control.

El bloc de memòria ROM guarda els pesos de 2·nbits de longitud que es necessiten per fer el processament. El bloc està format per un conjunt de 4 memòries ROM amb una profunditat de 48 paraules, ja que cada papallona necessita un pes. Cada etapa consta de 32 papallones (IFFT/FFT de 64 punts) i per tant cal guardar 192 pesos. El procés de generació dels coeficients es fa a través de Matlab a partir de les matrius de les etapes i després es carreguen a les memòries ROM.

La unitat de processament fa el càlcul de la IFFT. Es realitza un escombrat vertical de vuit períodes per a cadascuna de les 6 etapes. Per obtenir un resultat parcial de la unitat de processament es necessiten dos cicles de rellotge, un de lectura i un d'escriptura (Figura 4.28). El cicle de lectura proporciona a les papallones les dades i els pesos, i el cicle d'escriptura guarda el resultat parcial al bloc de memòria.

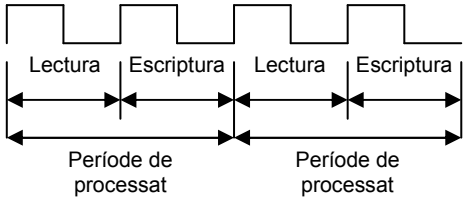


Figura 4.28 Període de procés

La unitat de processament està format per 4 papallones bàsiques amb una estructura idèntica a la de la Figura 4.19. Per dur a terme el càlcul d'una papallona cal un multiplicador i dos sumadors complexos, això equival a 4 multiplicadors i a 6 sumadors reals. Cal tenir en compte que els resultats parcials que s'obtenen durant el procés d'escombrat, es realimenten al bloc de memòries RAM de doble port. Per tant, es aconsellable enregistrar els diferents resultats parcials per superar les restriccions temporals (60MHz). Per no incrementar recursos en cada iteració, les sortides dels multiplicadors i sumadors, es trunquen al número de bits de l'entrada. També, existeix la possibilitat d'utilitzar blocs multiplicadors dedicats, per tal de millorar els resultats temporals.

El bloc de realimentacions s'encarrega d'encaminar els diferents resultats parcials obtinguts per la unitat de processament a les posicions adequades de les memòries, en funció de l'algorisme presentat. El bloc està format per 8 multiplexors de dues entrades de *nbits* controlats per la unitat de control.

El bloc de sortida extreu a 60MHz les dades de les 8 memòries RAM de doble port i les guarda en una altra memòria RAM de doble port a través del port A. Les dades guardades s'extreuen a 20 MHz pel port B i s'aprofita aquest procés per inserir el prefix cíclic. El bloc està format per una memòria RAM de doble port de 128 paraules de 2·nbits, i d'un multiplexor de 8 entrades de *nbits* per extreure les dades de les 8 memòries. El control d'aquest bloc, sincronitzat amb la unitat de control, està format per comptadors, registres i lògica combinacional. Per inserir el prefix cíclic, simplement s'utilitza el comptador d'adreces, el qual comença a llegir les dades a partir de les 16 últimes posicions de la memòria que corresponen al prefix cíclic.

b) Resultats

A les dues taules següents es mostren les característiques temporals dels blocs IFFT/FFT. El retard de 8μs és degut al paral·lelisme temporal dels processos d'adquisició i de processament.

Retard	Velocitat d'adquisició	Velocitat de processament	Velocitat D'extracció	Cicles de Processament + adquisició	Cicles de rellotge d'un període de processament
8µs	60MHz	60MHz	20MHz	48	2

Taula 4.8 Característiques temporals de la IFFT

Retard	Velocitat d'adquisició	Velocitat de processament	Velocitat D'extracció	Cicles de Processament + adquisició	Cicles de rellotge d'un període de processament
8µs	20MHz	60MHz	16MHz	48	2

Taula 4.9 Característiques temporals de la FFT

A continuació es presenten els resultats de síntesi per a una FFGA XCV2 2000. Per tal de poder fer una comparativa amb altres algorismes, es presenten els resultats en dues taules. La primera dóna els resultats de síntesi sense el procés d'extracció, ja que aquest inclou el procés d'inserció del prefix cíclic. La segona taula mostra els resultats de tot el bloc.

Long. paraula dades	Long. pesos	Slices	RAM	Mult 18x18	Màxima freqüència rellotge
14	14	834	10	16	84MHz

Taula 4.10 Recursos de la IFFT d'Uvic

Long. paraula dades	Long. pesos	Slices	RAM	Mult 18x18	Màxima freqüència rellotge
14	14	933	11	16	84MHz

Taula 4.11 Recursos de la IFFT i del bloc d'extracció d'Uvic

Els resultats obtinguts són bons, sobretot considerant que l'algorisme s'ha dissenyat amb un entorn d'alt nivell. Els recursos estan força optimitzats en comparació amb altres algorismes implementats a baix nivell [46][51]. La freqüència màxima de treball és de 84MHz tal com mostren les dues taules, superant la freqüència de 60MHz que especifiquen els estàndards. De totes maneres, es pot utilitzar l'atribut RLOC per ubicar millor els recursos de la FPGA des de l'eina de síntesi ISE per tal de millorar aquests resultats. També cal considerar que per estalviar recursos en les operacions de suma, resta i multiplicació, s'ha utilitzat l'opció *wrap-around* i l'opció de quantificació de truncament. Aquesta decisió comporta el perill de tenir saturació i menys precisió en les operacions.

Per altra banda, es mostren diferents simulacions de Simulink on es pot veure l'activitat de les adreces de les RAM de doble port de l'entrada i de la sortida.

La Figura 4.29 mostra l'activitat de les adreces dels dos ports de la RAM de doble port de l'entrada. Es pot veure la paginació que es va alternant en cadascun dels períodes. El port B, configurat sempre en mode escriptura, guarda les dades reordenant d'acord el *bit reversed*. El port A, té 3 fases de treball. A la primera, s'alterna la lectura amb l'escriptura calculant la FFT/IFFT. A la segona, de només de lectura, s'envia els resultats a la RAM de doble port de la sortida. A la tercera, el procés es manté inactiu.

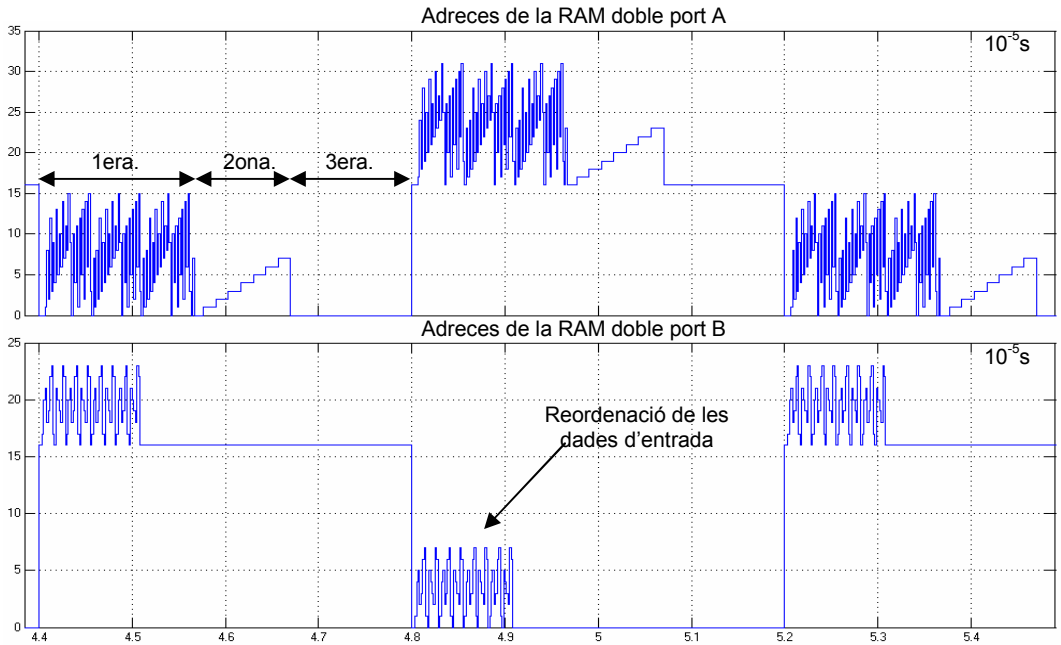


Figura 4.29 Adreces dels dos ports de la RAM doble port de l'entrada del bloc IFFT/FFT

La Figura 4.30 i la Figura 4.31 mostra l'activitat de les adreces dels dos ports de la RAM dual-port de sortida de la IFFT i la FFT respectivament. El port B, configurat sempre com a lectura, treu el resultat inserint el prefic cíclic pel cas de la IFFT (Figura 4.30), en canvi per la FFT només treu el resultat (Figura 4.31). El port A, rep les dades de la RAM de doble port de l'entrada i els reordena mentre els guarda.

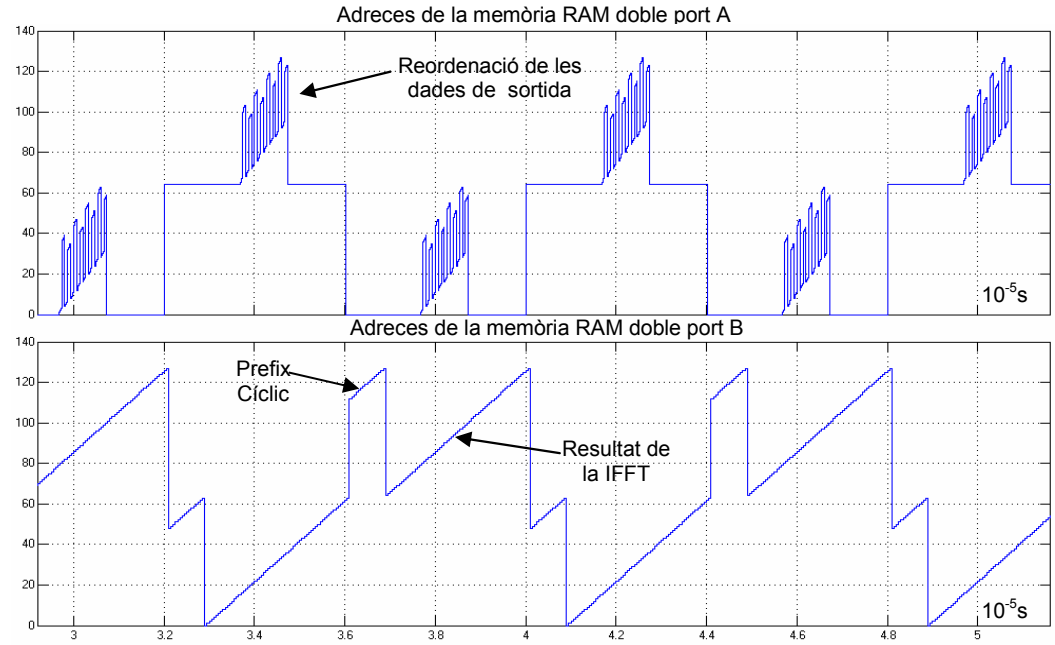


Figura 4.30 Adreces dels dos ports de la RAM doble port de la sortida de la IFFT

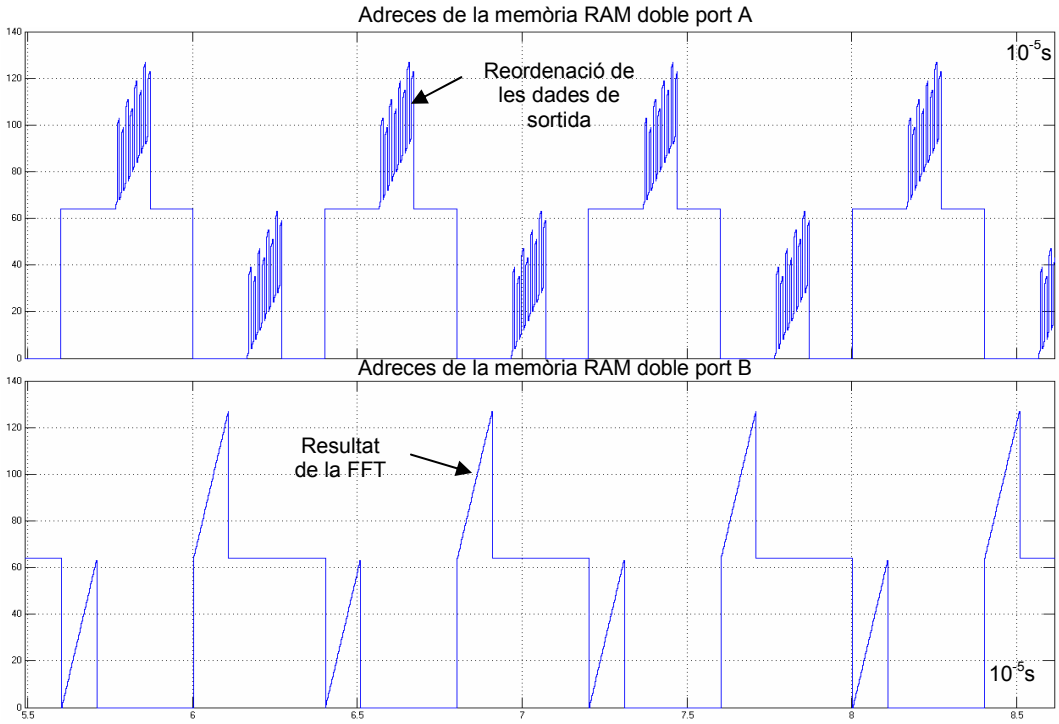


Figura 4.31 Adreces dels dos ports de la RAM doble port de la sortida de la FFT

c) Alternatives de disseny

Les alternatives de disseny utilitzant aquest mateix algorisme passen per l'exploració d'utilitzar més o menys papallones en l'element de procés. La Taula 4.12 mostra les diferents opcions possibles amb diferents cicles de rellotge i els recursos utilitzats de la FPGA. La restricció temporal més important és no sobrepassar el temps de símbol de 4μs. Aquest temps equival a 240 cicles de rellotge per a un processament a 60MHz.

Número de papallones	Cicles de rellotge de processament	Número de RAMs doble port	RAM doble port profunditat	ROM profunditat	Multiplexors
1	284+64	2	128	192	2
2	192+64	4	64	96	4
4	96+64	8	32	48	8
8	48+64	16	16	24	16

Taula 4.12 Exploració de recursos i cicles de l'element de processament

Segons els resultats mostrats a la taula només els elements amb 4 i 8 papallones compleixen amb els cicles de rellotge de processament mínims per fer el càlcul de la IFFT/FFT dins el temps de símbol OFDM. El valor que s'ajusta més en temps i recursos és l'opció de 4 papallones. Aquesta opció presenta l'inconvenient de no poder incrementar el paral·lelisme temporal, ja que llavors superaríem els 240 cicles. En canvi, en l'opció de 8 papallones, el paral·lelisme temporal es podria arribar fins a 4 cicles, però s'incrementen massa els recursos.

Una alternativa vàlida a aquest bloc, és la utilització de IPs IFFT/FFT (p.e. la de Xilinx V2.0). Aquesta opció s'ha comprovat en el model de l'emissor i receptor. Els resultats obtinguts són satisfactoris i per això s'ha fet una comparativa amb l'algorisme que s'ha presentat. Aquest bloc utilitza *dragonfly* (unitats bàsiques semblant a les papallones però de 4 entrades/sortides). La seva estructura es mostra a la Figura 4.32.

Les prestacions i recursos que ofereix aquest *core* per a la FPGA Virtex 2 xc2v3000 es poden veure a la següent taula:

Long. paraula dades	Long. pesos	Slices	RAM	Mult 18x18	Màxima freqüència rellotge	Cicles processament	Cicles adquisició + processament
16	16	1630	7	9	156MHz	96	161

Taula 4.13 Prestacions i recursos de la IFFT/FFT de Xilinx V2.0

Els multiplicadors d'aquesta taula són blocs multiplicadors HW dedicats (dels quals disposa la Virtex2). Cal destacar l'optimització de l'algorisme, ja que ofereix una màxima freqüència de rellotge de 156 MHz amb una ocupació de 1630 *slices*.

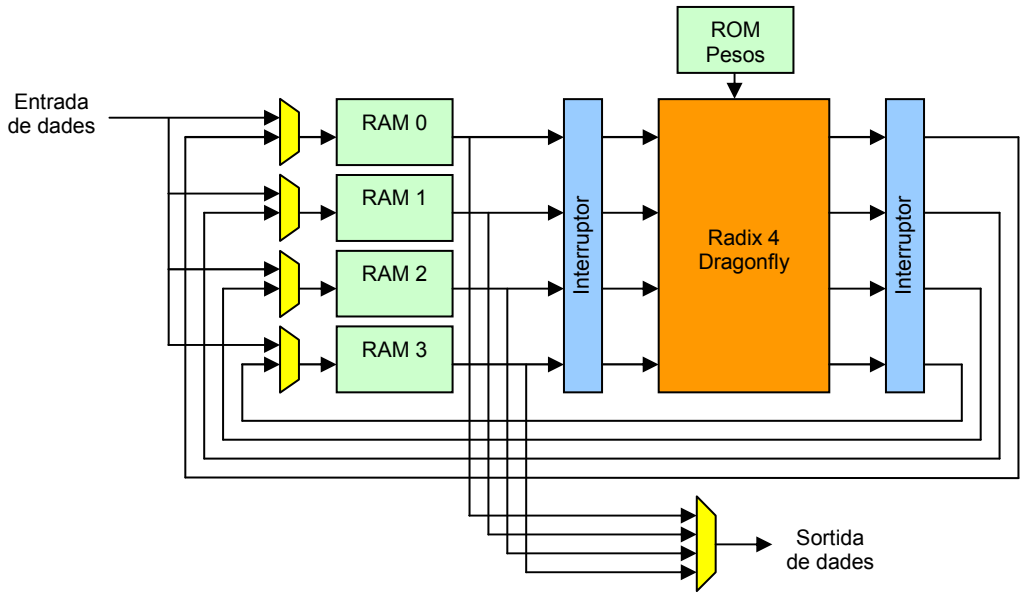


Figura 4.32 Estructura de la IFFT/FFT de Xilinx V2.0

4.3.6. Capçalera

Aquest bloc insereix un tipus de capçalera a l'emissor el principi de cada ràfega, en funció dels tipus de ràfega, especificat en l'estàndard HiperLAN/2. Aquests tipus de ràfega comentats al capítol 2, en concret en l'apartat de l'hiperLAN/2, són: control i difusió, enllaç descendent, enllaç ascendent amb capçalera curta i enllaç ascendent amb capçalera llarga.

Les especificacions de l'estàndard donen la composició d'aquestes capçaleres en el domini freqüencial. Per tant, utilitzant l'eina de Matlab, s'aplica la IFFT a aquestes capçaleres per obtenir el seu valor en el domini temporal. Aquests valors són guardats a dins una memòria ROM. Llavors només cal seleccionar la capçalera desitjada en funció del tipus de ràfega i síncronitzar-la amb les dades. En principi, la sincronització i la selecció es fixa per hardware.

a) Disseny

L'arquitectura d'aquest bloc es mostra a la següent figura, on cal destacar la unitat de memòria formada per dues ROM, una per la part real i l'altra per la part imaginària de la capçalera, amb la seva part de control formada per comptadors i lògica combinacional. A més, el bloc incorpora dos multiplexors, un per controlar la sortida de la capçalera o dades, i l'altre, per controlar la sincronització de la capçalera amb les dades. El bloc de sincronització insereix elements de retard i blocs de sincronització de mostreig, en funció del tipus de capçalera.

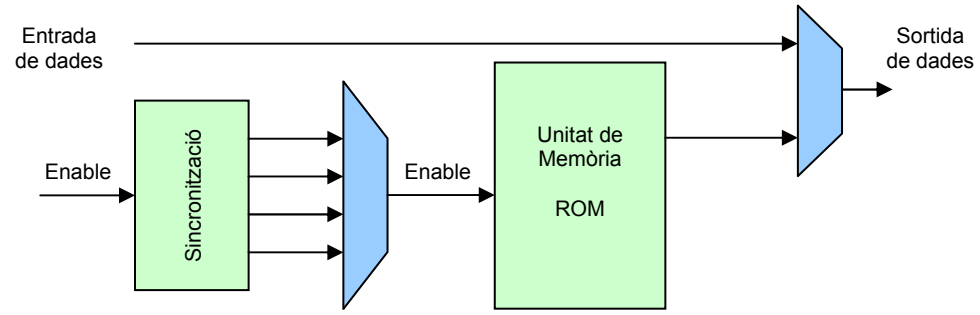


Figura 4.33 Estructura del bloc capçalera

b) Resultats

Els resultats de síntesi per aquest bloc es poden veure a les següents taules. La Taula 4.14 mostra els resultats dels recursos de la FPGA Virtex 2, implementant les memòries ROM amb memòria distribuïda mentre que la Taula 4.15 els mostra utilitzant blocs de memòria per a la implementació de les ROM.

Long. paraula dades	Slices	LUTs	freqüència rellotge
16	845	960	20MHz

Taula 4.14 Prestacions i recursos del bloc capçalera utilitzant memòria distribuïda

Long. paraula dades	Slices	LUTs	RAM	freqüència rellotge
16	61	96	2	20MHz

Taula 4.15 Prestacions i recursos del bloc capçalera utilitzant blocs de memòria

A continuació, la Figura 4.34 mostra la simulació de la capçalera curta per una ràfega d'enllaç ascendent en el domini temporal obtinguda des de Matlab/Simulink i System Generator. Es poden distingir les diferents seccions de les capçalera i la seva sincronització amb les dades.

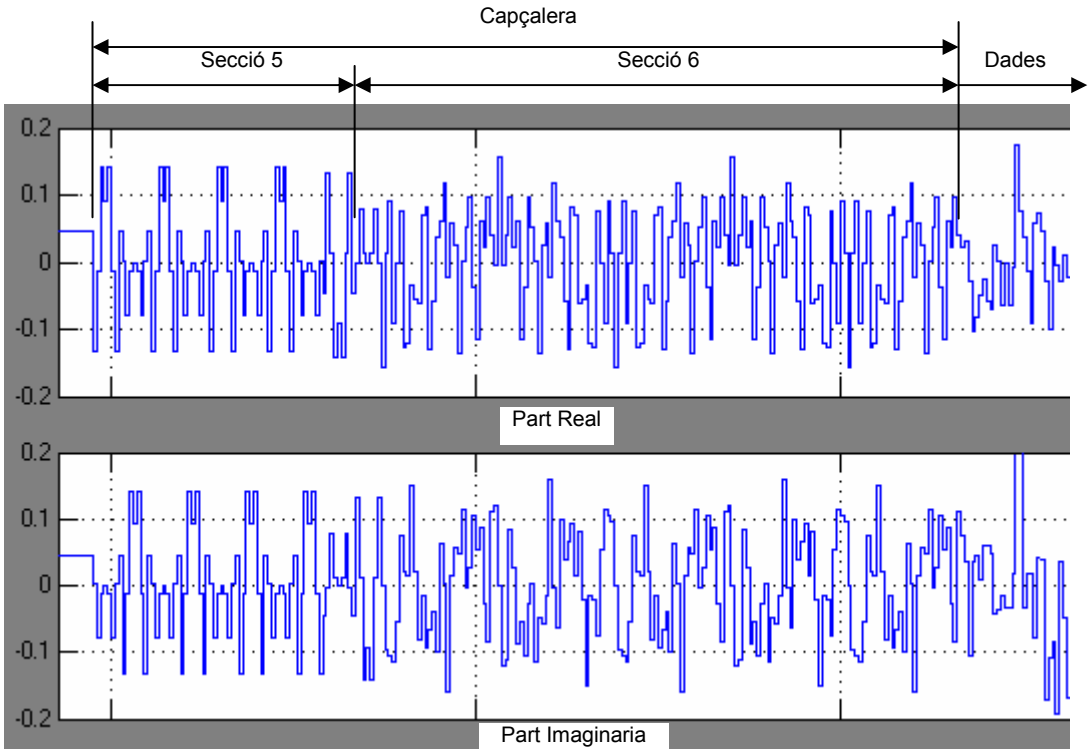


Figura 4.34 Domini temporal: capçalera per a una ràfega d'enllaç ascendent

c) Alternatives de disseny

Per les característiques força elementals d'aquest bloc, les alternatives passen per utilitzar més o menys memòries ROM. Es podria utilitzar una memòria ROM per a cada capçalera per fer més fàcil el control de la unitat de memòria. També, hi ha la possibilitat de concatenar la part real i la part imaginaria de la capçalera i utilitzar, només, una memòria ROM.

4.4. Receptor

La Figura 4.35 mostra el model el receptor HiperLAN/2 dissenyat. Es pot veure, en la cadena d'aquest emissor, els diferents blocs que el formen. Aquests blocs són lleugerament diferents dels de la cadena de l'emissor mostrada a la Figura 2.9, ja que els blocs s'han agrupat segons les necessitats de disseny i implementació.

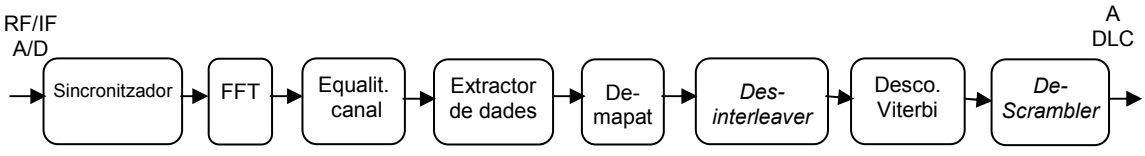


Figura 4.35 Cadena del receptor HiperLAN/2

4.4.1. Sincronitzador

La tècnica OFDM és efectiva per combatre les inclemències del canal. Una variació entre la referència freqüencial de l'emissor i el receptor, provoca una pèrdua d'ortogonalitat entre subportadores que pot originar una severa degradació en les prestacions del sistema. Per a la implementació pràctica és necessària una sincronització freqüencial gairebé perfecta [1].

En general, el procés de sincronització en freqüència es divideix en dues etapes. La primera és l'etapa d'adquisició i la segona l'etapa de seguiment que a la literatura sovint es referencien com *coarse synchronization* i *fine synchronization* respectivament. En l'àmbit d'OFDM durant l'etapa d'adquisició es redueix la desviació de freqüència inicial fins a la meitat d'aquest espaiat. Durant l'etapa de seguiment s'acaba de corregir la diferència residual.

Els algorismes que porten a terme el procés de sincronització es classifiquen en funció de si utilitzen pilots (dades conegudes tant per l'emissor com pel receptor) inclosos en el senyal rebut o bé si realitzen les estimacions de forma cega. En el primer cas es parla d'algorismes *data aided* i en el segon cas d'algorismes *non data aided* o *blind*.

Un altre aspecte que distingeix el procés de sincronització és el tipus de transmissió que es realitza, que pot ser contínua o bé orientada a paquets. Les tècniques usades en una i altra difereixen considerablement. L'hiperLAN/2 utilitza un tipus de transmissió orientades a paquets, per tant, només es considera aquest tipus.

Transmissió orientada a paquets

Per recuperar l'instant de temps i la desviació de freqüència a corregir, aquestes tècniques, utilitzen un o més símbols OFDM coneguts enviats al principi de cada ràfega. Aquests símbols es dissenyen per a tal propòsit. Les tècniques més comunes busquen l'inici de símbol a partir de mètodes de correlació. L'error d'estimació temporal es produeix quan la finestra de la FFT no encaixa amb la ideal (Figura 4.36). El principi de la finestra el marca l'inici del símbol o trama. Si aquesta es retarda, llavors, apart del propi error de sincronisme, s'afegeix l'efecte ISI (*Inter Symbol Interferency*) impossibilitant la recuperació del símbol OFDM. En canvi, si aquesta s'avança un número de mostres inferior a la diferència entre la longitud del prefix cíclic i de la resposta impulsional del canal, el símbol serà recuperable.

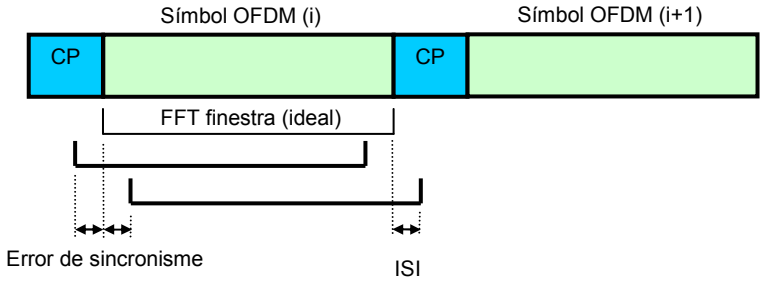


Figura 4.36 Efectes de l'error d'estimació temporal

La majoria dels mètodes de correcció de freqüència en mode de ràfegues realitzen la comparació de fase entre les dues meitats idèntiques del primer símbol d'entrenament enviat al principi de la ràfega. En aquest primer pas es proporciona una correcció fina del offset compresa entre ±0,5 de l'espai entre subportadores i a la que ens referirem com a part fraccionària del desplaçament de freqüència. L'error de freqüència resultant després de la correcció fina apareix com un múltiple sencer de la separació entre subportadores i el referirem com la part entera de la desviació de freqüència i, per tant, serà un múltiple enter de l'espai entre subportadores.

En la majoria dels mètodes la part entera de la desviació de freqüència és corregida utilitzant un segon símbol OFDM pilot. Del conjunt dels símbols pilot enviats al principi de cada ràfega també se'n extreu una estimació del canal.

L'article de referència en relació a aquests mètodes orientats a paquets es deu a Moose [52] qui proposa un estimador de màxima versemblança (*ML Maximum Likelihood*) per a la part fraccionària del desplaçament freqüencial amb un rang

d'adquisició limitat a ±1/2 de l'espai entre subportadores. En el mateix article es descriu com ampliar aquest rang reduint la longitud dels símbols d'entrenament.

Escurçant els símbols un factor 0,5 s'aconsegueix doblar el marge d'adquisició amb la contrapartida d'empitjorar la qualitat de l'estimació al disposar de menys mostres a promitjar. Per a un correcte funcionament, cal que la longitud dels símbols sigui major que la dels prefixes cíclics cosa que imposa una limitació a l'hora d'escurçar els temps de símbol. En Schmidl i en Cox [53] desenvolupen un mètode conjunt per a la detecció del temps i la freqüència que poden ser utilitzats tant en la transmissió de dades de forma contínua com a través de ràfegues i simplifica les necessitats de càlcul exigides a [52] introduint variacions al seu mètode i mantenint les mateixes prestacions que el seu estimador. Un cop corregida la part fraccionària del *doppler* s'estima la part entera correlant el primer símbol OFDM amb un segon símbol pilot després de la realització de la FFT. S'aconsegueix completar el sincronisme utilitzant una capçalera de només dos símbols OFDM.

Hi ha articles posteriors que adapten aquests mètodes per les capçaleres de HiperLAN/2 [54][56][57]. També hi ha un article interessant d'en S. Johansson i P. Nilsson [55] sobre la implementació d'un algorisme de sincronisme utilitzant el prefix cíclic basat amb l'article d'en M. Sandell i J. Van de Beek [58].

a) Disseny

A HiperLAN/2 durant la fase de control i difusió, el punt d'accés transmet una capçalera que es utilitzada pel terminal mòbil per realitzar la detecció del tipus de capçalera, sincronització en temps, sincronització en freqüència, control de guany automàtic i estimació de canal. La capçalera transmesa pel punt d'accés en la fase d'enllaç descendent està dissenyada per fer només l'estimació de canal. Finalment, les capçaleres transmeses pel terminal mòbil durant les fases d'enllaç ascendent i aleatòria han de permetre al punt d'accés, la sincronització en temps i en fase i l'estimació del canal (veure apartat 2.2.6. Tipus de ràfegues).

Abans de decidir quin mètode es vol utilitzar, cal comentar que les especificacions de l'estàndard HiperLAN/2, imposen que els oscil·ladors locals han de tenir una tolerància de ±20 parts per milió (ppm) [4]. Per tant, considerant el pitjor cas que seria treballar amb el canal més alt de freqüència 5,7GHz, amb una tolerància de ±40ppm (considerant la tolerància dels oscil·ladors locals del receptor i emissor), la variació màxima d'offset de freqüència que s'obté és de ±228KHz. Aquest valor està per sota de l'espai entre subportadores adjacents que és de 312.500Hz pel cas de 64 subportadores.

El mètode que s'ha aplicat [59], contempla la sincronització en temps i la sincronització de freqüència, deixant la detecció del tipus de capçalera i control de guany per a un treball futur. Aquest mètode, consta d'una etapa d'estimació de freqüència i d'una altra de sincronització en temps en el domini temporal. També seria necessari una etapa de seguiment d'offset de freqüència però que en aquest cas s'aplicarà en el domini freqüencial després d'aplicar la FFT.

Per tal de fer l'estimació de freqüència i sincronització temporal, s'ha escollit només un símbol, que és igual per totes les capçaleres i que correspon al següent símbol:

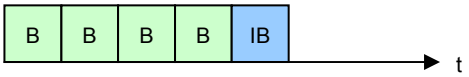


Figura 4.37 Símbol per fer la sincronització de freqüència i temps

S'ha escollit aquest símbol perquè està inclòs en totes les capçaleres on cal sincronitzar. Està composat per cinc seccions en el domini temporal, de les quals les quatres primeres són iguals i estan formades per 16 mostres. L'última secció, és igual

a les anteriors però de signe contrari. Aquest símbol encaixa amb l'algorisme d'en Schmidl-Cox, ja que les dues meitats del símbol, sense comptar el prefix cíclic són iguals. També, s'adapta al algorisme proposat per Hanzo-Keller [54] que consisteix en repetides còpies d'un patró de sincronització.

Per implementar l'estimador de freqüència s'ha optat per l'estimador d'en Hanzo-Keller, on el valor estimat de freqüència ve donat en el màxim de la funció de correlació $C(n_{max})$. Partint de les expressions d'en Hanzo-Keller [54], la funció de correlació ve donada per:

$$C(n) = \sum_{m=0}^{N+N_{CP}-N_S-1} x(n-m)\cdot x^*(n-m-N_S) \tag{14}$$

On $x(n)$ representen les mostres complexes rebudes, N és el nombre de subportadores per cada símbol OFDM, N_{CP} és la longitud del prefix cíclic, * representa el conjugat i N_S és la longitud de la periodicitat dels símbols.

I l'expressió d'estimació de freqüència per:

$$\Delta f_a = \frac{N}{N_S}\frac{F}{2\pi}\cdot ATAN\left[\arg_{\max}\left(C(n)\right)\right] \tag{15}$$

Si la longitud de la periodicitat dels símbols N_s s'agafa de 16 mostres, i sabent que N són 64 mostres, l'estimador de freqüència és capaç d'estimar un rang de desviació de $\pm 2F$, on F és l'espai entre subportadores. Aquest rang és suficient per cobrir les especificacions de l'estàndard HIPERLAN/2.

La següent figura mostra el diagrama de blocs corresponent a l'algorisme d'estimació de freqüència d'en Hanzo-Keller.

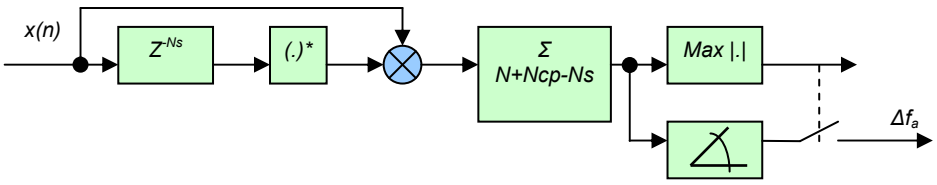


Figura 4.38 Diagrama de blocs de l'algorisme d'estimació de freqüència

Per a la sincronització temporal s'ha optat per l'algorisme proposat per en Hanzo-Keller (simplificació de l'algorisme Schmidl-Cox). La simplificació es basa en que només utilitzà la funció de la correlació $C(n)$ per obtenir el valor màxim que determina el començament de la trama.

L'expressió que presenta en Schmidl-Cox per mesurar la mètrica és:

$$M(n) = \frac{|C(n)|^2}{(P(n))^2} \tag{16}$$

On $C(n)$ i $P(n)$ són la funció de correlació i l'energia respectivament.

La següent figura mostra el diagrama de blocs corresponent a l'algorisme de sincronització temporal Hanzo-Keller. En aquest cas, la longitud de la periodicitat del símbol i del sumatori de mostres s'ha fixat a la meitat del símbol OFDM sense comptar el prefix cíclic ($L = 32$). D'aquesta manera, segons Schmidl-Cox [53], el fet de multiplicar el conjugat de la primera meitat del símbol per la segona meitat d'aquest cancel·la els efectes del canal, fent més robust el sincronisme temporal.

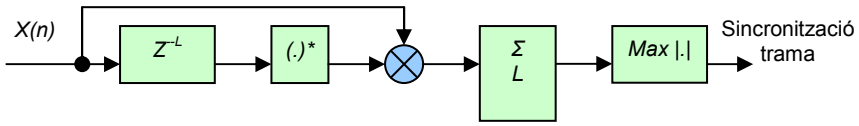


Figura 4.39 Diagrama de blocs de l'algorisme de sincronització temporal

Implementació del bloc de sincronisme temporal

El bloc de sincronització temporal és l'encarregat d'indicar al bloc de desmodulació FFT on comença el primer símbol OFDM. El primer símbol que ha de rebre el bloc FFT és el símbol C per fer l'estimació de canal. La següent figura mostra els blocs implementats per a la sincronització temporal corresponent a l'esquema funcional de la Figura 4.39.

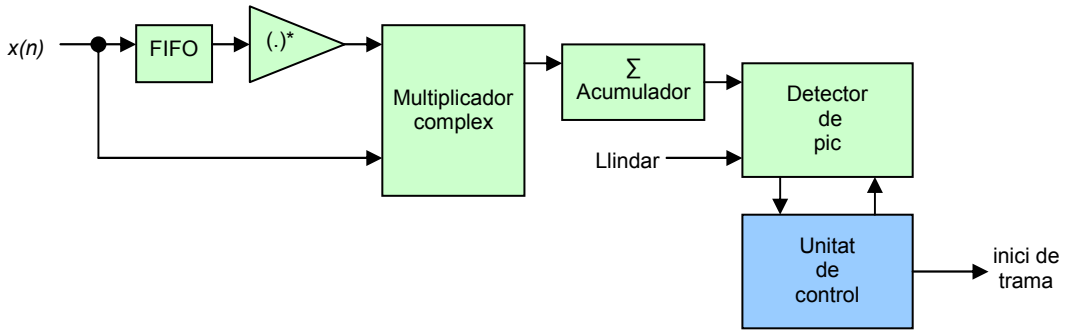


Figura 4.40 Diagrama de blocs per a la sincronització temporal

Els blocs FIFO, multiplicador complex i acumulador són els encarregats d'implementar la funció de correlació. Cal destacar el bloc acumulador que suma les parts reals i imaginàries per obtenir el mòdul acumulat (Figura 4.41). La longitud de la finestra de l'acumulador s'ha fet parametritzable (Z^{-L}) per tal de poder explorar altres algorismes.

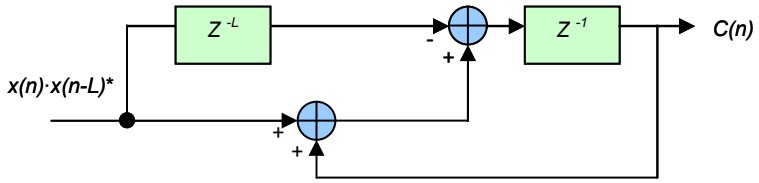


Figura 4.41 Bloc acumulador

La Figura 4.42 mostra l'algorisme implementat per detectar el màxim de la funció de correlació assignant un llindar. El valor d'aquest llindar s'ha obtingut de forma experimental a través de simulacions. Si el llindar és massa baix s'obtindria el màxim de la correlació abans del màxim "real", i si el llindar fos massa alt es podria donar el cas que en certes condicions (SNR, tipus de canal, *doppler*...) no superessin el llindar i no es trobés el màxim. En treballs futurs s'hauria d'associar aquest llindar amb el control automàtic de guany.

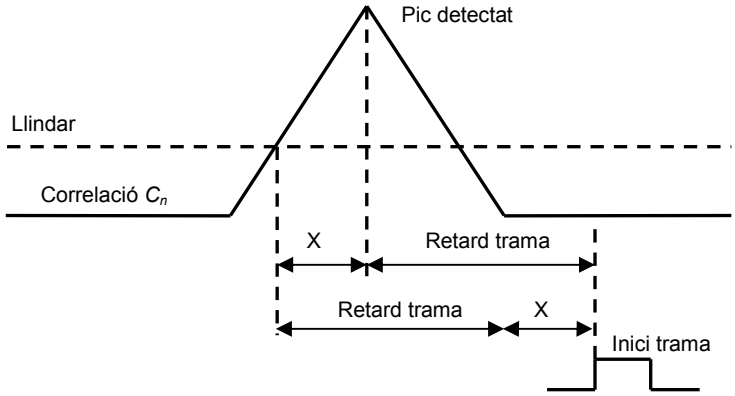


Figura 4.42 Detecció del pic i principi de trama

Un cop s'ha superat el llindar, es busca el pic comparant les mostres del valor absolut de la correlació amb el seu valor anterior. Si el resultat de la comparació és menor, vol que dir s'ha trobat el màxim de la correlació. Simultàniament a aquesta comparació, es compta la distància en mostres (X) del llindar fins al pic. El principi de sincronisme serà la suma entre el retard de trama i el valor de mostres comptades X a partir de la detecció del llindar. El retard de trama depèn del retard de les dades en passar pel bloc sincronitzador, el qual és funció de la implementació utilitzada (Figura 4.43). Aquest cas correspon a la següent expressió:

retard de trama = *buffer* de dades d'entrada + retard del CORDIC + longitud del prefix IB – inici de la finestra FFT.

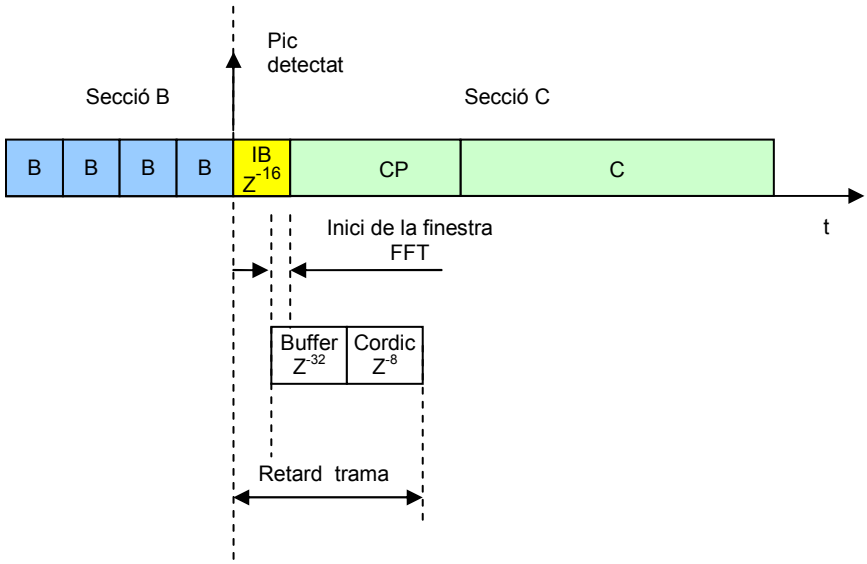


Figura 4.43 Retard de principi de trama respecte al punt de detecció del pic

El *buffer* de dades correspon a la FIFO d'entrada que retarda les dades, tal com mostra la Figura 4.40. El retard del CORDIC es deu a la unitat de CORDIC que disposa el bloc estimador de freqüència per corregir les dades en funció de l'offset de freqüència estimat. El prefix IB introdueix un retard de la seva longitud. L'inici de la finestra FFT és el marge de seguretat que s'introdueix per assegurar que estem a dins del mateix símbol OFDM.

La unitat de control s'encarrega de gestionar totes les fases d'aquest procés mitjançant una màquina d'estats finits tipus *Moore*. La Figura 4.44 mostra el seu diagrama d'estats. Cal destacar que l'estat d'entrada és l'estat 1 (espera de pic).

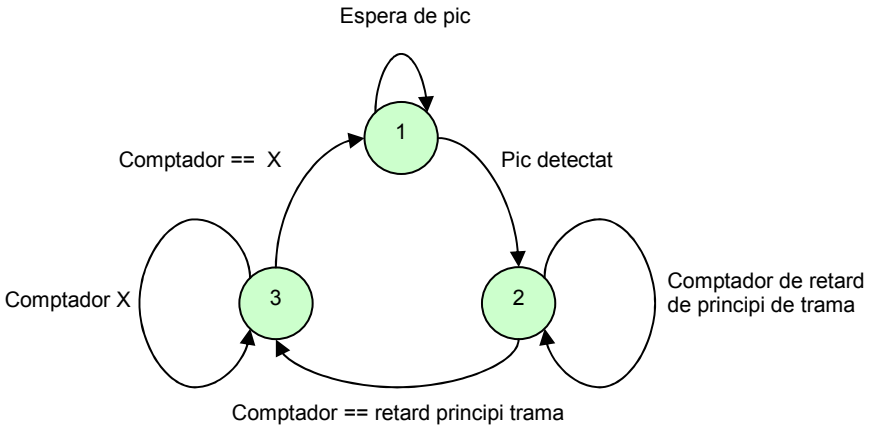


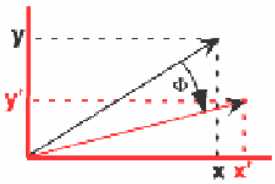
Figura 4.44 Diagrama d'estats finits de la unitat de control

Implementació del bloc de sincronisme freqüencial

Aquest bloc s'encarrega de fer l'estimació i correcció d'offset freqüencial. La Figura 4.45 mostra el diagrama de blocs del circuit estimador i corrector d'offset de freqüència. La sincronització en freqüència és semblant a la sincronització en temps, ja que s'utilitzen el mateixos blocs per fer la funció de correlació. La manera de detectar el pic és utilitzant un llindar (com en el cas anterior). Un cop trobat el valor màxim, s'estima l'*offset* freqüencial.

En la fase d'estimació es fa servir l'algorisme de CORDIC [60] per calcular l'offset de freqüència acumulat. L'algorisme CORDIC (*Coordinate Rotation Digital Computer*) permet el càlcul de funcions trigonomètriques d'una forma molt simple, ràpida, precisa i estalviant les multiplicacions. Té com a punt de partida la transformada rotació definida per Givens:

$$x' = x\cdot\cos\Phi - y\cdot\sin\Phi$$
$$y' = y\cdot\cos\Phi + x\cdot\sin\Phi \qquad (17)$$



Aquesta transformada, vàlida per qualsevol Φ, es particularitza per angles que compleixin: $\tan\Phi = \pm 2^{-i}$, per $i = 0,1,2\ldots$.Així s'aconsegueix que la multiplicació per la tangent quedi reduïda a una simple operació de desplaçament. L'algorisme de CORDIC es pot implementar amb una arquitectura paral·lela temporal o amb una arquitectura iterativa. L'arquitectura de paral·lelisme temporal utilitza més recursos que l'arquitectura iterativa i introdueix un retard inicial en funció de les etapes que disposa. En canvi, l'arquitectura iterativa utilitza menys recursos i introdueix un retard de processament en funció de les iteracions que es vulguin fer.

En el procés d'estimació de freqüència, només cal fer un càlcul per trama, per tant, es disposa de suficient temps per utilitzar una arquitectura iterativa. L'offset de freqüència estimat es guarda en un registre.

Una vegada s'ha estimat la freqüència, és necessari conèixer l'increment d'offset per a cadascuna de les subportadores. Aquest increment, és el resultat de dividir l'offset acumulat, estimat pel CORDIC, entre el valor de la longitud del *buffer Ns* (període repetitiu), segons l'equació (15).

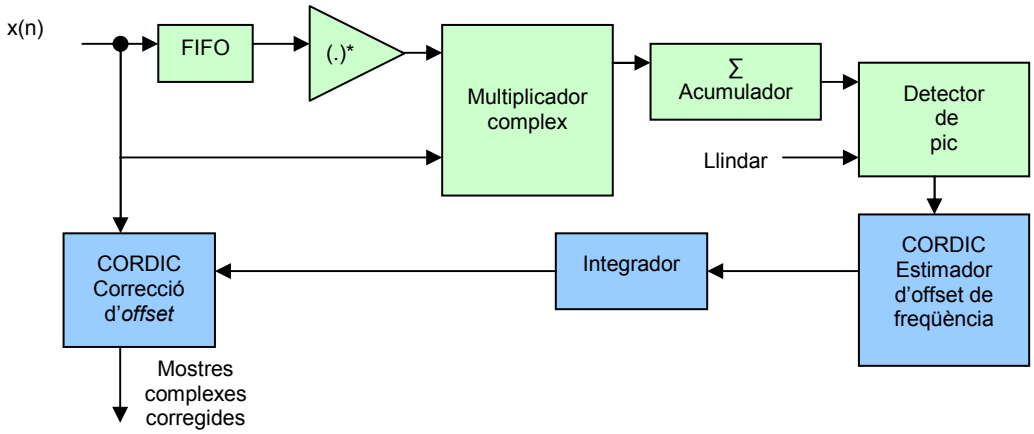


Figura 4.45 Sincronisme freqüencial: estimació i correcció de l'*offset* de freqüència

El bloc integrador va acumulant (en signe contrari a l'*offset* estimat per tal de compensar-lo) els increments d'*offset* estimats (en radians), i també acota a $\pm\pi/2$ truncant el valor acumulat.

Finalment, la fase de correcció va rotant les dades en temps reals en funció del valor que arriba del bloc integrador. Per fer aquesta rotació s'utilitza l'algorisme CORDIC amb l'arquitectura paral·lela, ja que és necessari corregir l'*offset* de freqüència de cadascuna de les dades. Cal tenir en compte que CORDIC només permet fer rotacions màximes de $\pm\pi/2$, per tant, per obtenir un CORDIC que permeti fer rotacions $\pm2\pi$, és necessari incloure una etapa prèvia que permeti situar la mostra en el quadrant correcte. Aquesta etapa està implementada amb lògica combinacional i multiplexors que detecten el quadrant. Llavors, mitjançant canvis de signe i/o canvis de la part real per la part imaginària es situen en el quadrant adient.

b) Resultats

A continuació es mostren les característiques temporals i de recursos de la FPGA utilitzats pel bloc de sincronització. Cal destacar que les dades arriben al bloc de sincronització a 20MHz.

La Taula 4.16 mostra els recursos del bloc sincronitzador format pels blocs de sincronisme temporal i freqüencial. Cal destacar que aquests resultats corresponen a una longitud de paraula de 16 bits (aquesta longitud de paraula pot variar en funció del procés de quantificació).

Long. paraula dades	Slices	LUTs	RAM	Mult 18x18	freqüència de treball	Retard del bloc
16	1350	2360	1	8	20MHz	42 Cicles

Taula 4.16 Recursos i característiques temporals del bloc de sincronització

Per tenir més detall de com es distribueixen el recursos del bloc de sincronització, es presenten per separat en dues taules: la Taula 4.17 pel bloc de sincronisme freqüencial i correcció, i la Taula 4.18 pel bloc de sincronisme temporal.

Cal destacar de la Taula 4.17 que els dos primers resultats de la taula són la comparació d'implementar de dues maneres (iterativa i paral·lela) l'algorisme CORDIC per calcular la fase acumulada de les mostres (*arctg*). Es pot veure que l'arquitectura iterativa implementada ocupa menys àrea (*slices*). Les longituds de paraula dels diferents blocs són diferents (venen determinades normalment pel bloc precedent) i s'han ajustat monitoritzant el rang dinàmic.

Les opcions de desbordament i de quantificació en les operacions de suma i resta que intervenen en els algorismes de CORDIC, s'han configurat per treballar en *wrap-around* i truncament per estalviar recursos. En canvi, al bloc acumulador s'ha mantingut la quantificació per truncament i s'ha canviat el *wrap-around* per l'opció de saturació. Habitualment, quan existeix una probabilitat de saturació no nul·la de l'aritmètica en determinades aplicacions, s'utilitza aritmètica saturant per evitar l'efecte *wrap-around*. Aquesta opció de desbordament per saturació evita efectes indesitjats en la detecció del pic.

Sincronisme freqüencial i correcció	Long. paraula dades	*Slices*	LUTs	RAM	Mult 18x18
CORDIC 8 Etapes arctg (iteratiu)	17	246	373	0	0
CORDIC 8 Etapes arctg (*Pipeline*)	17	302	514	0	0
Detector de Pic	16	47	37	0	0
Acumulador	18	163	286	0	0
Multiplicador Complex + *	16	84	168	0	4
CORDIC 8 Etapes Rotador	16	251	488	0	0
Integrador	16	122	220	0	0
FIFO	16	16	32	0	0

Taula 4.17 Recursos del bloc de sincronisme freqüencial

Sincronisme temporal	Long. paraula dades	*Slices*	LUTs	RAM	Mult 18x18
Unitat de Control		16	25	1	0
Detector de Pic	16	32	39	0	0
Acumulador	16	124	212	0	0
Multiplicador Complex + *	16	84	168	0	4
FIFO	16	16	32	0	0

Taula 4.18 Recursos del bloc de sincronisme temporal

c) Alternatives de disseny

Les alternatives de disseny d'aquest bloc poden ser moltes en funció dels algorismes que s'utilitzin per fer l'estimació de temps i freqüència. Una possible optimització al bloc dissenyat seria utilitzar el mateix bloc acumulador per a la sincronització temporal i freqüencial. També per evitar falsos pics, seria convenient posar un llindar de dues mostres per a cada cop que es detecti un possible màxim. D'aquesta manera es fan dues lectures a partir de la possible detecció del pic i només es validarà aquest quan dues d'aquestes lectures consecutives siguin més petites al possible pic .

4.4.2. Equalitzador de canal

L'equalitzador de canal corregeix la fase i l'amplitud del senyal rebut mitjançant uns coeficients que s'obtenen en el procés d'estimació del canal, comparant el senyal rebut amb una seqüència coneguda per poder estimar la distorsió de fase i l'atenuació d'amplitud del senyal.

Moltes de les tècniques d'estimació de canal són dissenyades tenint en compte que el canal varia en el temps i freqüència. Aquestes tècniques són especialment adequades per a transmissions continues com les *Digital Audio Broadcasting* o *Digital Video Broadcasting*.

En el cas de transmissions per ràfegues aquestes tècniques no són massa adequades per dues raons:

1.- En la majoria de la transmissions per ràfegues, com les *wireless* LAN, si la llargada de la trama és suficientment curta, es pot considerar que el canal és invariant en el temps mentre duri la trama.

2.- L'estimació utilitzant els pilots distribuïts en els símbols OFDM, introdueix un retard no desitjable de diversos símbols abans no es pugui fer la primera estimació. Aquest retard en la recepció de la trama produeix un retard en la resposta que fa decréixer l'efectivitat del sistema. A més, s'afegeix la necessitat d'emmagatzemar diversos símbols OFDM, per tant, incrementa el cost en recursos Hardware.

El mètode implementat considera el canal invariant en el temps i, per tant, podem fer l'estimació a partir de la capçalera coneguda que s'afegeix abans dels símbols OFDM, utilitzada també per l'estimació de sincronisme. D'aquesta forma s'obté l'estimació al principi i no s'introdueix cap retard. També, s'aprofiten les portadores pilots per fer un seguiment de fase, el qual permetrà corregir possibles desviaments en l'estimació d'offset de freqüència realitzada en l'etapa de sincronisme.

a) Disseny

L'equalitzador de canal s'ha separat en diferents parts tal com mostra el diagrama de blocs de la Figura 4.46. Cal destacar que l'equalitzador de canal està format bàsicament per quatre blocs: estimador de canal, inversor de coeficients de canal, corrector de dades i seguidor d'*offset* de freqüència.

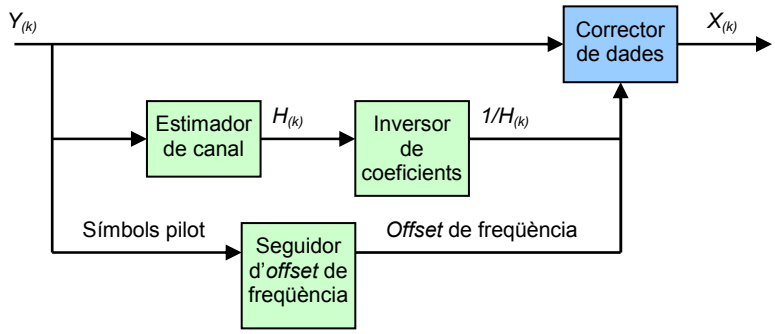


Figura 4.46 Diagrama de blocs de l'equalitzador de canal

Cal tenir en compte que és molt important cercar la millor alternativa d'implementació de cadascun dels blocs. També, és important tenir en compte el retard i número d'iteracions, ja que el sistema ha de corregir les dades a partir del primer símbol OFDM. Llavors l'estimador proposat, utilitza el mètode d'estimació Y/X (Y: dades rebudes, X: dades transmeses), el qual estima el canal utilitzant els símbols C de les capçaleres de HiplerLAN/2. Aquest mètode ofereix una computació ràpida i poc complexa i per tant, fàcilment implementable i s'ha escollit per la seva simplicitat [63][64].

Estimador de canal

Aquest bloc estima el canal a partir d'una seqüència o capçalera coneguda (pensada exclusivament per fer l'estimació de canal), formada per dos símbols OFDM *C* idèntics precedits per un prefix cíclic (*CP*), el qual és una còpia de les 32 mostres últimes del mateix símbol *C* (Figura 4.47).

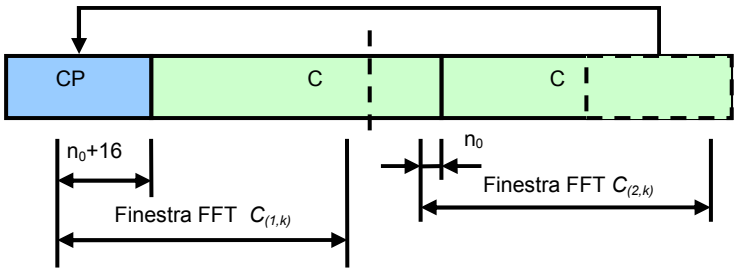


Figura 4.47 Seqüència d'estimació en el domini temporal

El símbol *C* es genera directament aplicant la IFFT a la següent seqüència:

$$SC_{-26..26} = \{1,1,-1,-1,1,1,-1,1,-1,1,1,1,1,1,1,-1,-1,1,1,-1,1,-1,1,1,1,1,1,0,1,-1,-1,1,1,$$
$$-1,1,-1,1,\ -1,-1,-1,-1,-1,1,1,-1,-1,1,-1,1,-1,1,-1,1,1,1,1\}$$

Funcionalment, es comparen les dades rebudes en el domini freqüencial amb la mitja de les dues parts que formen aquesta capçalera. El resultat d'aquesta comparació serà la resposta impulsional del canal H_k

Un cop s'ha aplicat la FFT a la seqüència d'estimació de la Figura 4.47, s'obtenen dos símbols $C_{(1,k)}$ i $C_{(2,k)}$ que es modelen com el producte del símbol d'estimació X_k i el canal H_k més un soroll additiu W_k.

$$C_{(i,k)} = X_k H_k + W_{(i,k)}, \quad \begin{cases} k = 0,1,2,\cdots,N-1 \\ i = 1,2 \end{cases} \tag{18}$$

$$\frac{C_{(i,k)}}{X_k} = H_k + \frac{W_{(i,k)}}{X_k} \tag{19}$$

La DFT presenta com a propietat el fet que un retard en el domini del temps equival a un angle de desfasament en el domini freqüencial, com mostra la següent equació:

$$x_{(n-n_0)} \leftrightarrow X_k e^{-j2\pi n_0 k/N} \tag{20}$$

Llavors els símbols d'estimació es poden expressar com:

$$C_{(1,k)} = C'_{(1,k)}\, e^{-j2\pi(n_0+16)k/N}, \quad k = 0,1,2,...N-1 \tag{21}$$

$$C_{(2,k)} = C'_{(2,k)}\, e^{-j2\pi n_0 k/N}, \quad k = 0,1,2,...N-1 \tag{22}$$

on $C'_{(i,k)}$ són els símbols rebuts després del procés de la FFT sense soroll. L'angle de desfasament degut al retard de mostres n_o més les 16 mostres s'inclou en l'estimació del canal i, per tant, serà compensat quan s'equalitzin les dades, donant així més flexibilitat a la sincronització temporal.

Per altra banda, segons les equacions (21) i (22) els dos símbols $C_{(1,k)}$ i $C_{(2,k)}$ tenen una estructura matemàtica igual però amb un angle de desfasament acumulat de π/2, ja que un desfasament de 16 mostres sobre un total de 64 mostres (un símbol OFDM), equival a 90º. Llavors, per fer la mitjana entre els dos símbols, cal "rotar" incrementalment 90º el símbol $C_{(1,k)}$. S'aplica una seqüència cíclica de 0º, 90º, 180º i 360º per anar rotant els 90º totes les 64 les subportadores. Aquesta rotació es podria fer amb un CORDIC, però el cost associat seria molt elevat. Una manera més simple

és fer una mapat intercanviant part imaginària per part real i els seus signes de la següent manera:

- Rotar 90º : intercanvi de les parts real i imaginària, i canvi de signe de la part real.
- Rotar 180º: canvis de signe de les parts real i imaginària.
- Rotar 270º: intercanvi de les parts real i imaginària, i canvi el signe de la part imaginària.
- Rotar 0º o 360º: No fer res.

S'ha dissenyat un mòdul format per una unitat de control i una unitat de procés (Figura 4.48). La unitat de control disposa d'una entrada d'habilitació (*enable*) i de tres sortides (*SignR, SignI i Rotate*). La seqüència que genera la unitat de control i que s'envia a la unitat de procés per fer el mapat es pot veure a la Taula 4.19.

SignR	*SignI*	*Rotate*	Angle
0	0	1	0º
1	0	0	90º
1	1	1	180º
0	1	0	270º
0	0	1	360º

Taula 4.19 Seqüència de rotació de 90º

La unitat de procés està formada bàsicament per multiplexors que permeten fer l'intercanvi de les parts reals per les parts imaginàries i el seu signe en funció de la seqüència rebuda de la unitat de control.

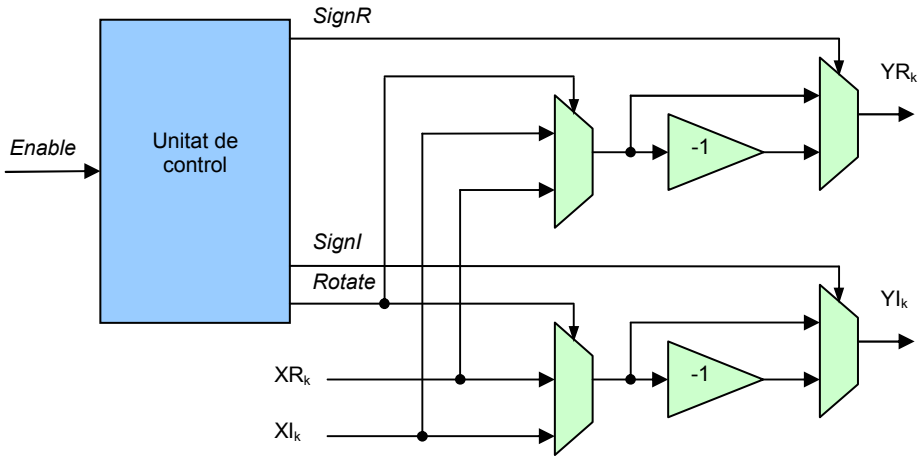


Figura 4.48 Sistema de procés per fer una rotació incremental de 90º

Llavors el canal estimat es calcula amb la següent expressió:

$$\hat{H}_k = \frac{C_{(1,k)} e^{j2\pi 16k/N} + C_{(2,k)}}{2X_k} \tag{23}$$

$$\hat{H}_K = H_k + \frac{W_{(1,k)} + W_{(2,k)}}{2X_k} \tag{24}$$

Els sorolls gaussians $W_{(1,k)}$ i $W_{(2,k)}$ són estadísticament independents, així la variància de la seva suma dividida per dos és la meitat de la variància dels sorolls gaussians individuals.

A continuació, agafant l'amplitud del símbol d'estimació X_k igual a 1, l'equació (23) es pot expressar de la següent manera:

$$\hat{H}_{(1,k)} = \frac{C_{(1,k)} e^{\frac{j2\pi 16k}{N}}}{X_k} = \begin{cases} C_{(1,k)} e^{\frac{j2\pi 16k}{N}} & \text{si} \quad X_k \geq 0 \\ -C_{(1,k)} e^{\frac{j2\pi 16k}{N}} & \text{si} \quad X_k < 0 \end{cases} \tag{25}$$

$$\hat{H}_{(2,k)} = \frac{C_{(2,k)}}{X_k} = \begin{cases} C_{(2,k)} & si \quad X_k \geq 0 \\ -C_{(2,k)} & si \quad X_k < 0 \end{cases} \tag{26}$$

Per tant, Els resultats $\hat{H}_{(1,k)}$ i $\hat{H}_{(2,k)}$ són els mateixos símbols $C_{(1,k)}$ i $C_{(2,k)}$, tenint en compte els signes del símbol conegut X_k.

Per implementar les equacions (25) i (26) s'utilitza una memòria ROM on guardem els signes del símbol conegut X_k (0 manté el signe, 1 canvia el signe). Aquesta memòria adreçada per un comptador, va generant els bits que controlen els signes dels símbols $C_{(1,k)}$ i $C_{(2,k)}$ a través de multiplexors (Figura 4.49).

Finalment, només falta fer la mitjana:

$$\hat{H}_K = \frac{\hat{H}_{(1,k)} + \hat{H}_{(2,k)}}{2} \tag{27}$$

En el cas de la implementació de la mitjana (27), el símbol $\hat{H}_{(1,k)}$ es retarda 64 mostres per tal de sincronitzar temporalment $\hat{H}_{(1,k)}$ i $\hat{H}_{(2,k)}$ i després se sumen. El resultat de la suma es divideix per dos (desplaçament a la dreta d'un bit) i es guarda en una memòria RAM que s'utilitzarà posteriorment per obtenir els coeficients inversos del canal (Figura 4.49).

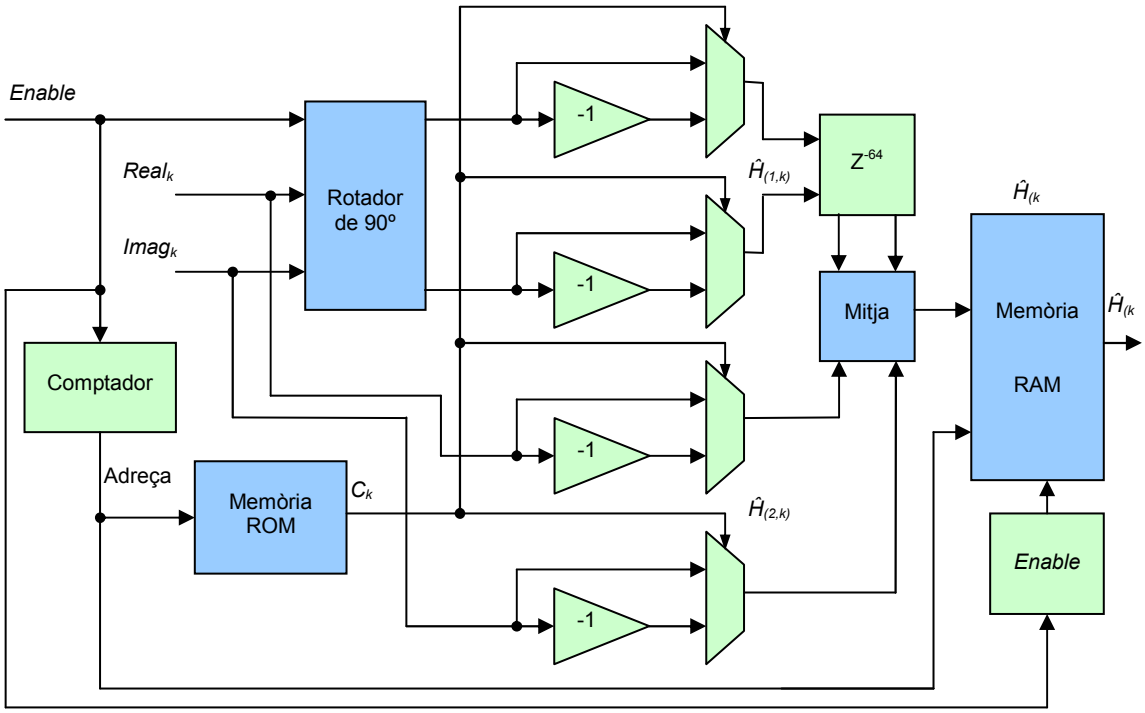


Figura 4.49 Diagrama de blocs de l'estimador de canal

Inversor dels coeficients de canal

El resultat de l'estimació de canal $\hat{H}_k$ té la forma complexa $a_k + b_k j$, per tant, cal buscar el seu mòdul i fase per tal de facilitar la divisió i obtenir el coeficients inversos de canal.

$$\frac{1}{\hat{H}_k} = \frac{1}{a_k + b_k j} = \frac{1}{\left|\hat{H}_k\right| \cdot e^{j\hat{\alpha}}} = \frac{e^{-j\hat{\alpha}}}{\left|\hat{H}_k\right|} \tag{28}$$

Per obtenir el mòdul i la fase s'utilitza l'algorisme CORDIC (*COordinate Rotation DIgital Computer*) [60] i per fer l'invers del mòdul s'utilitza el mètode de Hung [61].

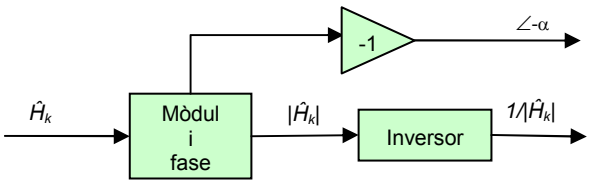


Figura 4.50 Inversor dels coeficients de canal

L'algorisme de CORDIC ja s'ha explicat anteriorment en el bloc de sicnronisme, per tant, a continuació s'explicarà el mètode utilitzat per fer l'invers del mòdul.

El càlcul de l'invers d'un nombre, i per tant dels possibles algorismes de divisió, és una operació important per a moltes àrees de computació: processament de senyal, aplicacions numèriques i científiques. Oberman i Flynn [62] han explorat els principals algorismes i la seva implementació, i presenten una taula comparativa dels algorismes de divisió (*SRT, Newton-Raphson, series expansion i accurate quotient aproximation*), donant per a cadascun els recursos, retards i iteracions necessàries (Taula 4.20). Posteriorment, en Hung i el mateix Flynn [61] mostren un algorisme ràpid per fer la divisió utilitzant una petita memòria (*Small Lookup Table*). Els principals avantatges que presenta el mètode de Hung respecte als algorismes citats anteriorment, és que no necessita iterar, té un retard de 2 cicles de rellotge i utilitza dos multiplicadors i una petita memòria. A més, per fer l'invers s'estalvia un dels multiplicadors.

Algorismes i mètodes		**Retard**	**# d'iteracions**	**Capacitat de la Memòria (LUT)**
SRT Radix-4		12	12	--
Newton Raphson	8b seed	6	3	128B
	16b seed	4	2	64KB
Series expansion	8b seed	7	3	128B
	16b seed	5	2	64KB
Accurate quotient approximation		3	1	19.5KB
Hung		2	0	13KB

Taula 4.20 Comparació de diferents algorismes de divisió

Per tant, permet fer l'invers de cadascuna de les dades que arriben, sense necessitat d'iterar, i la seva implementació HW és senzilla.

Mètode de Hung

El mètode de Hung [61] és un algorisme nou de *high radix* que es basa en les expansions de les series de Taylor. Els algorismes previs consideraven per separat cadascun dels termes individuals en les series de Taylor, donant com a resultat una *look-up table* molt gran. En canvi, aquest algorisme proposa considerar conjuntament cadascun del termes individuals en les series de Taylor, reduint considerablement la *lookup table*.

Pel nostre cas, l'entrada a l'invers serà sempre positiva perquè aquesta és el mòdul $|H_k|$. Després de realitzar varies simulacions amb les diferents possibilitats dels canals

de HiperLAN/2 i utilitzant histogrames, s'ha previst un rang final de 0 a 4. Per tant, es necessiten dos bits enters més els bits decimals que s'ajustaran posteriorment.

Tenint en compte aquests paràmetres i partint d'un número *Y* de 2m-bit entre zero i aproximadament quatre, definit per la següent expressió:

$$Y = 2^{1}y_0 + 2^{0}y_1 + 2^{-1}y_2 + \cdots + 2^{-(2m-2)}y_{2m-1}, \text{ on } y_i \{0,1\} \quad (29)$$

Per calcular 1/Y, es descomposa Y en dues parts: Y_h i Y_l. On Y_h conté els *m* bits més significatius i Y_l conté els *m* bits menys significatius, tal com mostren les dues següents expressions:

$$Y_h = 2^{1}y_0 + 2^{0}y_1 + 2^{-1}y_2 + \cdots + 2^{-(m-2)}y_{m-1} \quad (30)$$

$$Y_l = 2^{-(m-1)}y_m + 2^{-m}y_{m+1} + \ldots + 2^{-(2m-2)}y_{2m-1} \quad (31)$$

Per exemple, per un número *Y* de *2m*=14, el rang de Y_h aniria entre 0 i 3.9688 ($4 - 2^{-(m-2)}$) i el rang de Y_l aniria entre 0 i 0.0310 ($2^{-(m-2)} - 2^{-(2m-2)}$).

D'acord amb aquestes expressions, per calcular *1/Y* es parteix de la següent equació:

$$\frac{1}{Y} = \frac{1}{Y_h + Y_l} = \frac{Y_h - Y_l}{Y^2{}_h - Y^2{}_l} \quad (32)$$

A continuació, utilitzant les series de Taylor, l'equació (32) s'expandeix tal com mostra la següent equació:

$$\frac{1}{Y_h + Y_l} = \frac{1}{Y_h}\left(1 - \frac{Y_l}{Y_h} + \frac{Y^2{}_l}{Y^2{}_h} - \cdots\right) \quad (33)$$

Combinant els dos primers termes de la equació (33) s'arriba a la següent aproximació:

$$\frac{1}{Y} \approx \frac{Y_h - Y_l}{Y^2{}_h} \quad (34)$$

A la Figura 4.51 es pot veure el mètode de Hung implementat utilitzant l'equació (34). Els blocs són un restador, una memòria i un multiplicador. Cal tenir en compte que per l'aplicació que s'està fent, el multiplicador no pot tenir un retard superior a un cicle de rellotge.

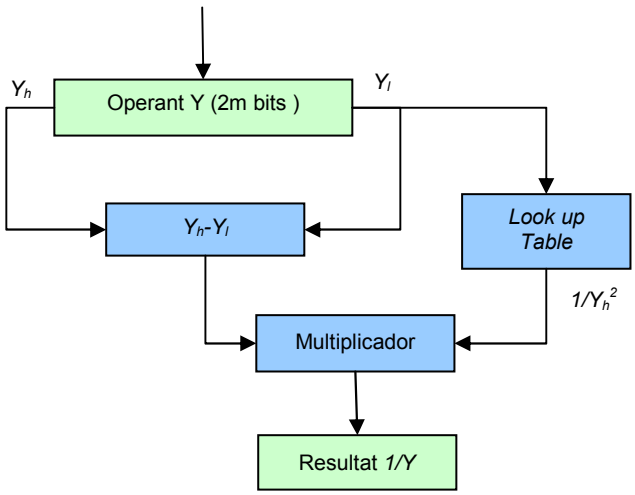


Figura 4.51 Divisió 1/Y: Mètode de Hung

En aquest mètode hi ha tres fonts d'error [61]: error d'aproximació en les sèries de Tailor E_∞, error d'arrodoniment de l'amplada de la paraula de la memòria E_T i error d'arrodoniment del multiplicador E_M. Llavors l'error total és la suma dels tres errors $E_\infty+E_T+E_m$ (Figura 4.52). Cal esmentar que els errors E_T i E_m són mínims en comparació amb l'error E_∞ i es poden minimitzar augmentant el número de bits.

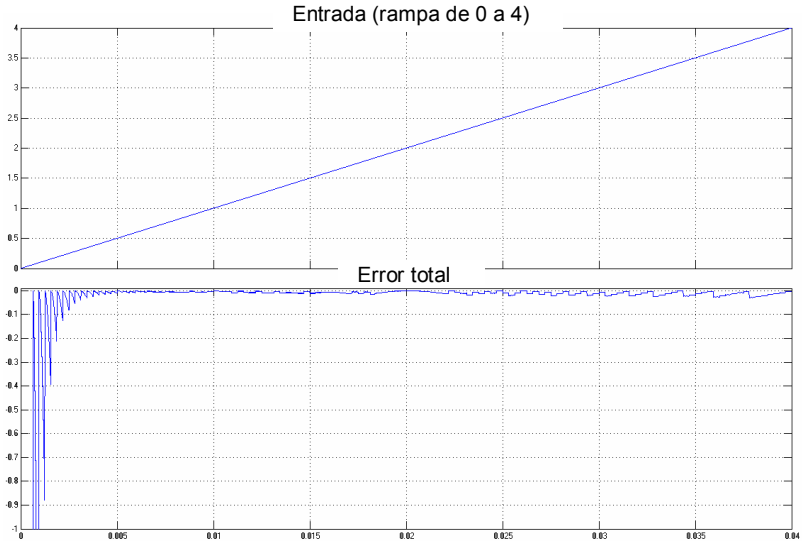


Figura 4.52 Error total de la divisió 1/Y

Perquè funcioni correctament aquesta implementació s'ha d'emplenar la memòria de forma correcta amb els valors $1/Y^2_h$ tenint en compte els rangs de treball de Y_h. A continuació es pot veure com s'obtenen aquests valors a través de Matlab.

1/sum(binariza(a).[2^-5 2^-4 2^-3 2^-2 2^-1 2^0 2^1])^2)*, on *a = 1,2,3,4,..., 127* (35)

Per minimitzar l'error de arrodoniment de l'amplada de paraula de la memòria (E_T), es passa a 2m+2bits. Llavors, considerant m = 7 bits, la capacitat de la memòria serà de 128 posicions i tindrà una amplada de paraula de 16 bits.

Seguidor d'offset de freqüència

Aquest bloc corregeix el desfasament acumulat per símbol OFDM, a causa de l'error d'estimació d'*offset* realitzat en l'etapa anterior de sincronisme. Per calcular aquest desfasament es poden utilitzar el pilots que s'insereixen en els símbols OFDM. Aquests pilots són generats a partir del polinomi x^7+x^4+1 i modulats amb BPSK (-1, 1) en l'emissor d'OFDM.

Per saber el desfasament, simplement s'ha d'obtenir la fase d'aquests pilots, tenint en compte el signe de la part real. Això només és vàlid, si aquesta correcció és petita i el desfasament no supera els ± π/2. La correcció es podria fer utilitzant els quatre pilots, o en menys, en funció de quantes vegades per símbol OFDM es volgués fer la correcció.

El desfasament acumulat és molt petit i s'ha decidit fer una sola correcció per cadascun dels M símbols OFDM ($n = 1,2...M$), agafant com a referència l'últim pilot de cada símbol ($P_{1,n}$). Aquest desfasament que s'obté s'acumula i s'acota truncant a ±2π. No es descarta en el futur la possibilitat d'utilitzar totes les portadores pilot per millorar el seguiment.

El desfasament acumulat obtingut, se li canvia el signe i posteriorment es passa al bloc corrector de dades, el qual farà la correcció oportuna.

La Figura 4.53 mostra el seu diagrama de blocs, on cal destacar el bloc de CORDIC que calcula el desfasament. Aquest bloc de CORDIC s'ha implementat amb una

estructura iterativa. Per aquest cas en concret, el número d'iteracions és de 8 provocant un retard de 13 cicles de rellotge. Aquest retard no porta problemes ja que el desfasament acumulat és molt petit i, per tant, si es comença a corregir l'offset 13 cicles més tard l'efecte és mínim.

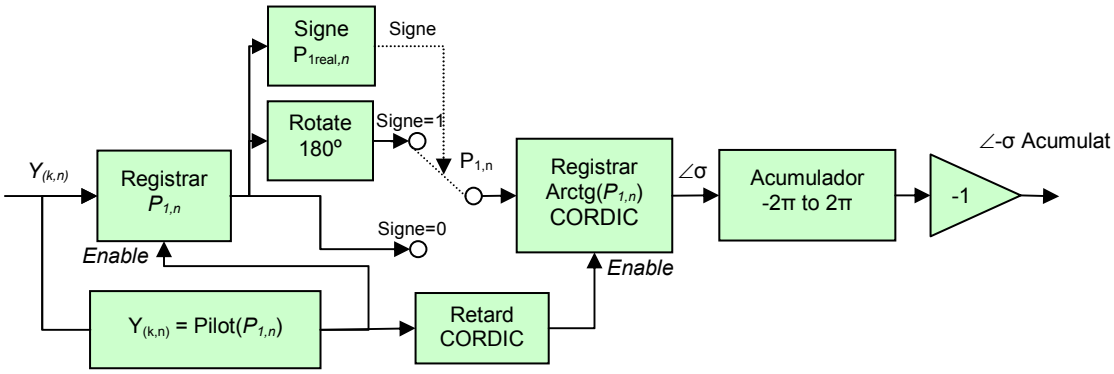


Figura 4.53 Diagrama de blocs del seguidor d'offset de freqüència

Corrector de dades

El bloc corrector de dades corregeix la distorsió en fase i l'atenuació d'amplitud del senyal que li arriba en funció dels coeficients inversos de canal obtinguts. A més, també corregeix el desfasament acumulat obtingut a través del bloc de seguidor d'offset de freqüència, tal com mostra la següent equació:

$$X_{(k,n)} = \frac{1}{\left|\hat{H}_k\right|}.Y_{(k,n)}.e^{-j(\hat{\alpha}_k+\sigma_n)} \qquad (36)$$

On $\hat{\alpha}_k$ i $\hat{H}_k$ són la fase i el mòdul estimats del canal a la portadora k i σ_n és el desfasament acumulat del símbol OFDM n del bloc seguidor d'*offset* de freqüència.

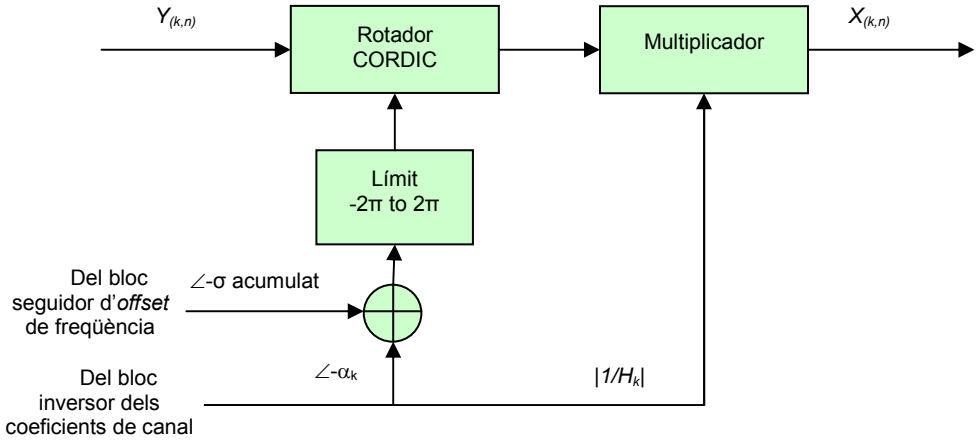


Figura 4.54 Corrector de dades

A la Figura 4.54 es mostra el diagrama de blocs del corrector de dades, on cal destacar el sumador que suma les dues fases, distorsió de fase estimada $\hat{\alpha}_k$ i desfasament acumulat σ_n, donant com resultat la fase total a corregir. Aquest resultat s'acota trucant a ±2π.

L'algorisme de CORDIC només permet fer una rotació de ±π/2. Llavors cal inserir una etapa de mapat prèvia que permeti cobrir els ±2π per aconseguir un rotador complet. El CORDIC implementat segueix una estructura de paral·lelisme temporal, ja que s'ha d'anar corregint en temps real. En aquest cas no és possible implementar

l'estructura iterativa perquè suposaria un retard per a cadascuna de les dades a corregir.

b) Resultat

Per obtenir resultats de l'equalitzador de canal, s'ha inclòs a dins d'un model HiperLAN/2 amb emissor i receptor. El model utilitzat té les següents característiques: 12 Mbit de velocitat de transmissió, modulació QPSK, relació de codificació ½, 48 subportadores, 4 símbols pilots, 16 mostres de prefix cíclic, 4μs de període de símbol OFDM i un rellotge de 60MHz.

El rellotge del bloc equalitzador és de 16 MHz, d'aquesta manera es manté un flux de dades contínues provinents del bloc FFT. Les restriccions temporals s'han fixat a 50ns.

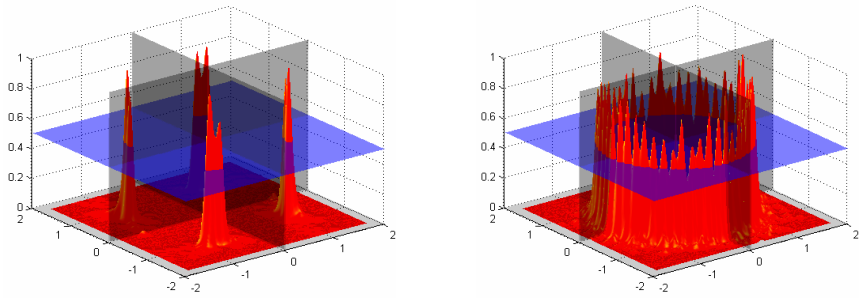
És molt important seleccionar la precisió correcta dels senyals del bloc equalitzador, amb l'objectiu d'obtenir un balanç òptim entre prestacions i cost. A la Taula 4.21, l'estimador Uvic es refereix al nostre disseny utilitzant l'eina System Generator (utilitzant punt fix) i, l'estimador Simulink, es refereix al mateix model validat amb Simulink (utilitzant coma flotant).

Bits de precisió	$\hat{H}_k$ error max. (%)	$1/\hat{H}_k$ error max.(%)	$Y_{(n,K)}/\hat{H}_k$ error max.(%)
10	0.96	13.60	15.98
12	0.38	6.25	6.04
14	0.11	2.44	2.45
16	0.01	2.03	2.02

Taula 4.21 Error relatiu en funció de l'estimador Uvic respecte a l'estimador Simulink que treballa en coma flotant

Aquesta taula dóna una primera aproximació dels bits de precisió necessaris per al bloc equalitzador. Llavors, un cop es tingui tot el model, s'haurien d'ajustar tots els senyals dels blocs, establint un llindar de bits erronis i mitjançant simulacions de tota la cadena, obtenir la precisió de bits més òptima que no superi aquest llindar establert.

Per comprovar el bloc seguidor, s'han fet diverses simulacions per obtenir la distribució de la constel·lació QPSK, tal com mostra la Figura 4.55.

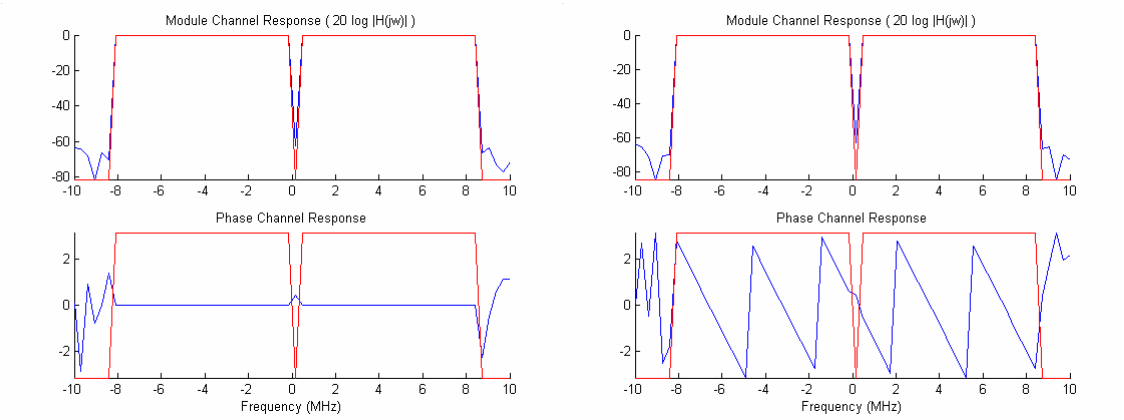


a) b)

Figura 4.55 Distribució de la constel·lació: a) amb seguidor d'offset de freqüència b) sense seguidor d'*offset* de freqüència

La Figura 4.55 a) mostra la distribució de la constel·lació QPSK utilitzant el bloc seguidor d'*offset*. Cal destacar que es manté en una posició estàtica perquè s'està corregint el petit error de desfasament que s'hagi produït en la etapa de sincronització. En canvi, la Figura 4.55 b) mostra la distribució de la mateixa constel·lació però sense utilitzar el bloc seguidor d'*offset*. Es pot apreciar perfectament a la figura que en aquest cas la constel·lació gira perquè no s'ha compensat el petit error de desfasament produït en l'etapa de sincronització.

S'ha comprovat l'estimador sense canal ni soroll i variant el retard de principi de trama tal com mostren les següents figures.

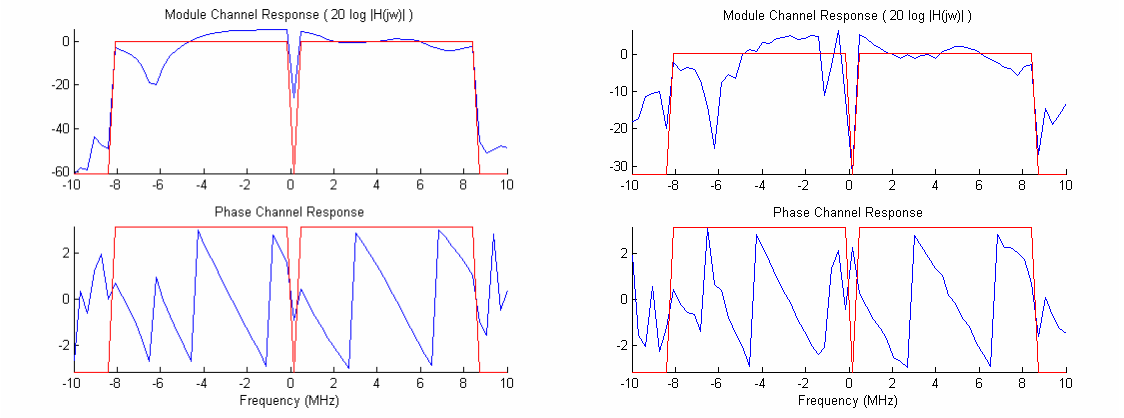


a) b)

Figura 4.56 Canal estimat sense canal ni soroll: a) retard de començament de trama n_0=0 i b) retard de començament de trama n_0=6

Es pot veure a la Figura 4.56 a) que el canal estimat és correcte, ja que aquest no ha sofert cap atenuació ni distorsió de fase del senyal rebut en el rang de freqüències de treball perquè no s'ha utilitzat cap canal ni soroll en la simulació i s'ha fixat un retard de començament de trama de '0'. També cal destacar a la Figura 4.56 b) que la fase estimada amb un retard de començament de trama diferent de '0' no és constant, ja que un retard en el domini temporal equival a un desfasament en el domini freqüencial.

També s'han fet simulacions amb els cinc canals de HiperLAN/2, amb i sense soroll i amb i sense *doppler*. A les següents figures es mostra una simulació pel cas del canal A estimat amb i sense soroll gaussià.



a) b)

Figura 4.57 Canal A estimat amb un retard de començament de trama de n_0=5 i *doppler*=0: a) SNR=60dB i b) SNR=10dB

A la Figura 4.57 es pot veure el resultat de l'estimació del canal A amb i sense soroll gaussià. Es pot apreciar perfectament les atenuacions i distorsions de fase del senyal rebut en el rang de freqüències de treball.

La Taula 4.22 mostra els recursos totals de l'equalitzador i de cadascun dels blocs que el formen utilitzant 14 bits de precisió. Cal tenir en compte que tots el operadors que permeten l'ajust de quantificació i de desbordament s'han ajustat per truncament i

per ajust de desbordament respectivament per tal d'estalviar recursos. Aquesta mesura implica tenir menys precisió i més risc de desbordament.

14 bits	Equalitzador	Estimador de canal	Inversor de coeficients	Seguidor d'offset	Corrector
Slices	1295	186	367	243	490
BRAMs	1	1	0	0	0
FFs	1191	129	412	193	448
LUTs	2218	354	618	368	869
Multiplicadors	3	0	1	0	2

Taula 4.22 Recursos de l'equalitzador per a una FPGA Virtex II 2000

Segons els resultats presentats en aquesta taula el corrector és el que utilitza més recursos, ja que necessita un CORDIC per fer la rotació a cadascuna de les dades que arriben i que per tant, l'estructura d'aquest CORDIC no pot ser iterativa. També s'utilitzen dos multiplicador dedicats per tal d'estalviar recursos. En canvi, en el seguidor d'offset de freqüència el CORDIC té una estructura iterativa, ja que es disposa de més temps per fer el càlcul.

Els resultats de recursos de la FPGA són bons en comparació amb altres algorismes implementats amb metodologies semblants amb la mateixa FPGA. Per exemple, els resultats són un 41% millors en comparació als resultats presentats en el llibre *OFDM Wireless LANs: a theoretical and practical guide* [3].

c) Alternatives de disseny

Les alternatives de disseny poden ser moltes depenent de l'algorisme que s'utilitzi per equalitzar. Com s'ha comentat anteriorment, s'ha buscat un algorisme senzill i fàcil d'implementar que lògicament té alternatives de disseny. Per exemple per fer l'invers també es podria haver utilitzat l'algorisme de CORDIC. El problema més gran que presenta aquest algorisme és que necessita bastants recursos, tal com s'ha pogut comprovar amb els resultats presentats al bloc de sincronització i en aquest mateix bloc.

També cal considerar el bloc seguidor d'*offset* que s'ha triat l'opció de fer una correcció per símbol OFDM i es podria haver implementat amb l'opció de fer fins a 4 correccions per símbol, una per a cadascun dels símbols pilot, però aquesta opció s'ha descartat perquè el desfasament que s'havia de corregir era massa petit.

4.4.3. Extractor de dades

Aquest bloc té la funció contrària al bloc símbol OFDM. Llavors el bloc extractor de dades elimina el símbols pilot, la component de contínua i les subportadores laterals que arriben conjuntament amb les dades i, a més, torna a deixar les dades en el mateix ordre que estaven abans d'arribar al bloc símbol OFDM.

a) Disseny

Lògicament el disseny d'aquest bloc és molt semblat al bloc símbol OFDM però fent les accions contràries. Per tant, l'element principal és una memòria de doble port que fa de *buffer*, el qual permet adaptar les velocitats d'entrada/sortida i ordenar les dades. S'ha fet una paginació de la memòria en dues pàgines: pàgina 0 i pàgina 1 per tal de mantenir un flux continu de dades. Les dades reals i imaginàries són concatenades i guardades en una mateixa posició de memòria. Llavors hi ha una unitat de control, formada bàsicament per un comptador i lògica combinacional, la qual és habilitada pel senyal de dada vàlida que envia el bloc equalitzador, que controla l'habilitació del punter d'escriptura. Aquesta habilitació permet l'escriptura sempre i quan no hi hagi un símbol pilot, la component contínua o les dades corresponents a les bandes lateral. En

canvi el punter de lectura s'habilita un cop quan s'ha acabat escriure la pàgina 0. Per tant entren 64 dades i en surten 48. La Figura 4.58 mostra els seu diagrama de blocs.

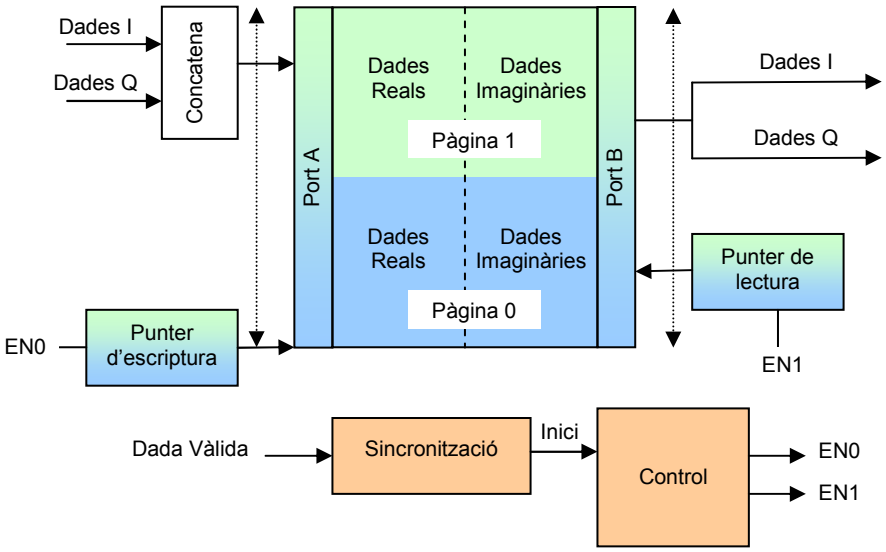


Figura 4.58 Diagrama de blocs de l'extractor de dades

b) Resultats

A la següent figura es pot veure l'evolució temporal dels punters d'escriptura i lectura. Cal destacar els punts on el punter d'escriptura és deshabilitat per tal de no gravar els símbols pilot, la component continua i les dades de les bandes laterals. A més, es pot veure que el punter de lectura va escombrant de forma contínua les dues pàgines. També, cal destacar que quan el punter d'escriptura escriu a la pàgina 0, el punter de lectura llegeix la pàgina 1.

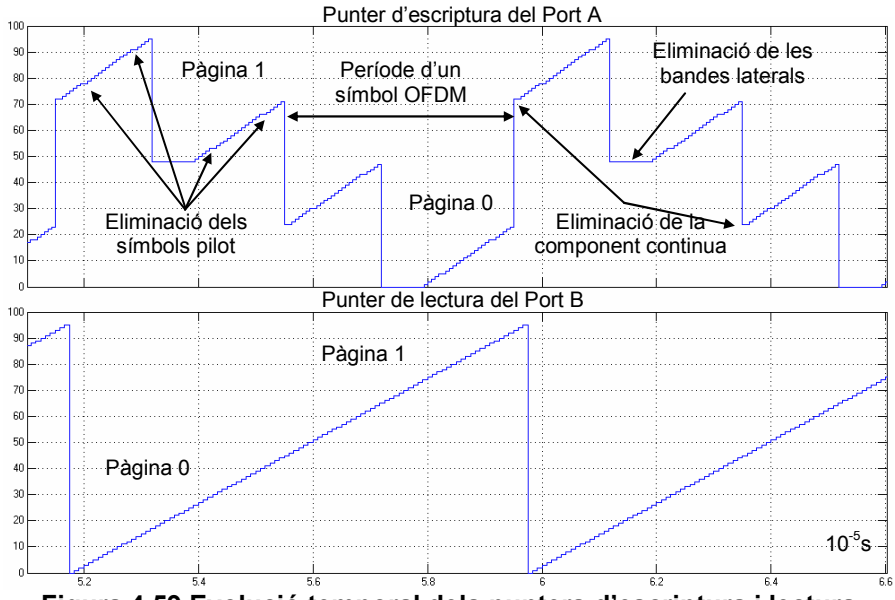


Figura 4.59 Evolució temporal dels punters d'escriptura i lectura

A la següent taula es presenten els resultats on la majoria dels recursos utilitzats són deguts als comptadors dels punters d'escriptura i lectura i de la unitat de control.

FPGA Virtex II 2000	
Slices	83
BRAMs	1
FFs	55
LUTs	107
Multiplicadors	0

Taula 4.23 Recursos de FPGA del bloc extractor de dades

c) Alternatives de disseny

Aquest bloc és força senzill i les possibles alternatives de disseny són les mateixes comentades en el bloc símbol OFDM.

4.4.4. Desmapat

La funció d'aquest bloc és desmapar els símbols que li arriben en un format adequat pel descodificador convolucional. L'estàndard fixa un descodificador convolucional de Viterbi de 64 estats amb decisió *soft.* Per un descodificació en decisió *soft* l'entrada consisteix en un codi binari format per més d'un bit que representa la fortalesa o debilitat dels "0" i dels "1". Llavors és necessari que el desmapat descodifiqui directament els símbols al codi binari establert. La Figura 4.60 mostra la constel·lació QPSK, on cal destacar a la part esquerra i de baix de la figura com canvien els bits *b2* i *b1* en funció del quadrant.

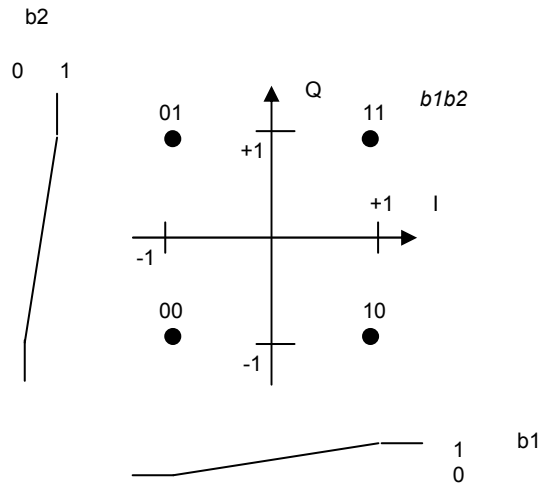


Figura 4.60 Constel·lació QPSK a desmapar

La descodificació del QPSK només depèn dels signes dels símbols *I* i *Q* tal com mostra la següent expressió:

$$b_1 = sign(I)$$
$$b_2 = sign(Q) \qquad (37)$$

El resultat que s'obté d'aquesta descodificació són directament els bits *b1* i *b2* i no un codi binari. Per tant, cal modificar-la per tal d'obtenir una descodificació que doni directament el codi binari que es necessiti. La Taula 4.24 mostra un exemple de codificació en dos formats diferents utilitzant 4 bits. La conclusió que s'arriba és que serà necessari saber en quin quadrant de la constel·lació està el símbol a descodificar. En cas que el símbol estigui en una zona que el bit pot passar de "1" a "0" o l'inrevés, s'haurà de quantificar en N nivells en funció dels bits que formin el codi. Finalment, només faltarà codificar el codi d'acord amb la quantificació.

	Magnitud amb signe	Magnitud sense signe
"1" fort	1111	1111
	1110	1110
	1101	1101
	1100	1100
	1011	1011
	1010	1010
	1001	1001
"1" dèbil	1000	1000
"0" dèbil	0000	0111
	0001	0110
	0010	0101
	0011	0100
	0100	0011
	0101	0010
	0110	0001
"0" fort	0111	0000

Taula 4.24 Format de les dades pel Viterbi *Soft*

a) Disseny

El codi que s'envia al Viterbi per indicar el grau de fortalesa i de debilitat dels "1" o "0", és el codi binari sense signe que es mostra a la Taula 4.24. El diagrama d'aquest descodificador, corresponent a la part real, es pot veure a la Figura 4.61. La principal funció és detectar en quina zona està el *símbol(I)* a descodificar. Si la zona que està el *símbol(I)* el valor del bit *b1* no varia, s'assigna a la sortida un valor fix de "15" o "0" depenent del valor del bit ("1" o "0") mitjançant un multiplexor. Per detectar aquesta zona simplement s'utilitzen dos comparadors i una *and* lògica. En cas contrari, el símbol(I) es quantifica en 16 nivells, ja que s'utilitzen 4 bits pel codi. Per quantificar es determina el rang a través d'una resta i el resultat s'escala al fons d'escala desitjat, en aquest cas a 15, multiplicant per 7.5 mitjançant tres desplaçaments i una resta.

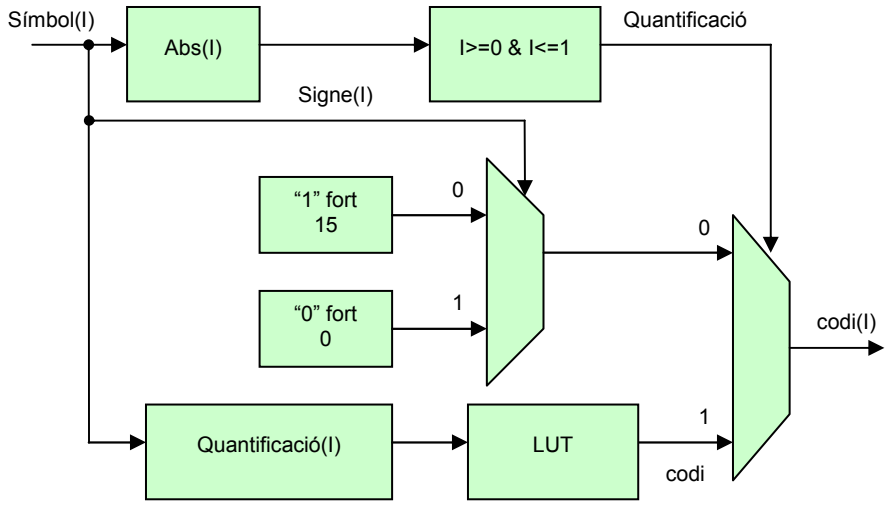


Figura 4.61 Descodificador QPSK de la part real

b) Resultats

Els recursos que utilitza aquest bloc són mínims ja que només fa servir multiplexors, comparadors i memòria. S'ha de tenir en compte que s'han concatenar els símbols reals i imaginaris i que llavors s'han de treure en sèrie. La taula següent mostra els recursos utilitzats de la FPGA per a la seva implementació.

FPGA Virtex II 2000	Memòria distribuïda	Blocs de memòria
Slices	211	203
BRAMs	0	2
FFs	27	19
LUTs	384	376
Multiplicadors	0	0

Taula 4.25 Recursos de FPGA utilitzats del bloc desmapat QPSK

La Taula 4.25 mostra els recursos implemementant la LUT utilitzant memòria distribuïda i blocs de memòria dedicats. Per la implementació s'ha utilitzat la memòria distribuïda ja que la LUT és de 16x4 i no interessa gastar dos blocs de memòria per implementar una LUT tan petita.

La Taula 4.26 mostra els recursos utilitzats per un desmapat de 16QAM seguint la mateixa estructura de la Figura 4.61. Es pot veure que els recursos es tripliquen amb respecte al desmapat QPSK, degut a que hi ha més zones a detectar en la constel·lació 16QAM.

FPGA Virtex II 2000	Memòria distribuïda
Slices	627
BRAMs	0
FFs	119
LUTs	1099
Multiplicadors	0

Taula 4.26 Recursos de FPGA utilitzats del bloc desmapat 16QAM

4.4.5. *Deinterleaver*

Els bits desmapats codificats en paraules de 4 bits són reordenats en dos passos. Aquests dos passos són el contrari de les permutacions fetes en el bloc de l'interleaver.

Si s'anomenen *k* i *i* els índexs d'ordenació d'entrada i de sortida de la primera permutació respectivament, la primera permutació es pot expressar amb la següent equació:

$$i = s\lfloor k/s \rfloor + (k + \lfloor k/N_{CBPS} \rfloor) \bmod s, \qquad k = 0,1,...,N_{CBPS} - 1 \tag{38}$$

On N_{CBPS} és el número de bits codificats en un símbol OFDM, el símbol $\lfloor\ \rfloor$ indica l'arrodoniment per baix de la part sencera, *mod* és el residu de la divisió i *s* es defineix com: $s = \max(N_{BPSC}/2,1)$.

En les modulacions BPSK o QPSK aquesta primera permutació no afecta ja que el número de bits per subportadora (N_{BPSC}) valen 1 o 2.

Pel cas de la segona permutació, si s'anomenen *i* i *j* els índexs d'ordenació d'entrada i de sortida de la segona permutació respectivament. Llavors, la segona permutació es pot expressar amb la següent equació:

$$j = 16 \cdot i - (N_{CBPS} - 1)\left(\lfloor 16 * i / N_{CBPS} \rfloor\right), \qquad i = 0,1,...,N_{CBPS} - 1 \tag{39}$$

a) Disseny

El bloc *deinterleaver* és idèntic al bloc *interleaver* dissenyat amb la diferència dels comptadors que intercanvien els seus valors per tal de canviar les files per columnes i

a l'inrevés. També hi ha la diferència que al bloc *interleaver* permuta bits i aquest paraules de quatre bits. Això no implica cap canvi en el disseny ja que només afecta a l'amplada de la paraula del bloc memòria. Per tant, el diagrama de blocs del *deinterleaver* és el mateix que mostra a la Figura 4.5 del bloc *interleaver*.

b) Resultats

Els mòdul *deinterleaver* obtingut és parametritzable en funció del número de bits per subportadora, facilitant la seva futura integració a l'emissor d'OFDM en funció del mode seleccionat.

El retard del bloc serà el número de bits codificats per símbol OFDM (N_{CBPS}) + 1 cicle de rellotge *($(N_{BPSC}$*16*3)+1)*, tal com mostra la següent taula.

N_{CBPS}	Retard (Cicles de Rellotge)
48	49
96	97
192	193
288	289

Taula 4.27 Retard del *deinterleaver* en funció dels bits codificats per símbol OFDM

La següent taula mostra els recursos utilitzats per bloc *deinterleaver* en funció dels bits codificats per símbol OFDM, utilitzant una FPGA Virtex 300.

N_{CBPS}	48	96	192	288
SLICEs	31	38	47	58
BRAMs	1	1	1	1

Taula 4.28 Recursos de FPGA del bloc *deinterleaver*

4.4.6. Viterbi

Les dades codificades amb el codificador convolucional poden ser descodificades utilitzant l'algorisme de Viterbi desenvolupat per Andrew J. Viterbi l'any 1967 [65].

El procés de descodificació avalua una mètrica que compara un conjunt consecutiu de dades rebudes, amb totes les possibles combinacions vàlides que poden agafar aquest conjunt i, n'estima la realització més probable consistent amb l'observació feta.

Una manera de representar i explicar aquest procés de descodificació és utilitzant el diagrama de Trellis.

La descodificació es fa mitjançant una simplificació del *Maximum Likelihood Decoder* (ML), que selecciona, per definició, el valor estimat de la sortida *y* del descodificador que maximitza la probabilitat d'haver transmès una entrada *r* concreta condicionada a aquesta sortida *P(r/y)*.

Segons l'algorisme de Viterbi, per reduir el número de càlculs, cada cop que dos camins s'uneixin en un estat en el diagrama de Trellis, el de major mètrica acumulada es desestima en la recerca del camí òptim.

S'ha trobat experimentalment que longituds d'exploració del diagrama de Trellis ≥ 5K (K=M+1, veure apartat 4.3.1 Codificador Convolucional), ja no aporten millores substancials al procés de descodificació [66]. Cal tenir en compte, que el descodificador aconsegueix millors resultats quan utilitza codis de convolució òptims. Per tant, l'estàndard HiperLAN/2 ha fixat aquest codi per a una relació de codificació de ½ [4].

a) Disseny

Per implementar aquest bloc s'ha partit d'un bloc de Viterbi dissenyat a la Universitat de Vic [67][68][69]. L'arquitectura utilitzada per dissenyar el descodificador

convolucional basat en l'algorisme de Viterbi es fonamenta en el diagrama de Trellis i amb el seu mecanisme de descodificació. El descodificador inclou les següents unitats bàsiques tal com mostra la Figura 4.62:

Addició Comparació Selecció (ACS).

Aquest mòdul inclou per a cadascun dels estats del diagrama de Trellis, una operació per acumular les mètriques de cadascuna de les branques, una operació de comparació entre els diferents camins que convergeixen en un mateix node i una operació de selecció del camí amb millor mètrica.

Acumulador de camins (AdC).

Permet guardar els camins supervivents per a cadascun dels estats. Per tant, disposa de memòria.

Màxima versemblança (MV).

Determina el camí de màxima versemblança a partir de la millor mètrica obtinguda al final de cada trama i determina el codi de sortida.

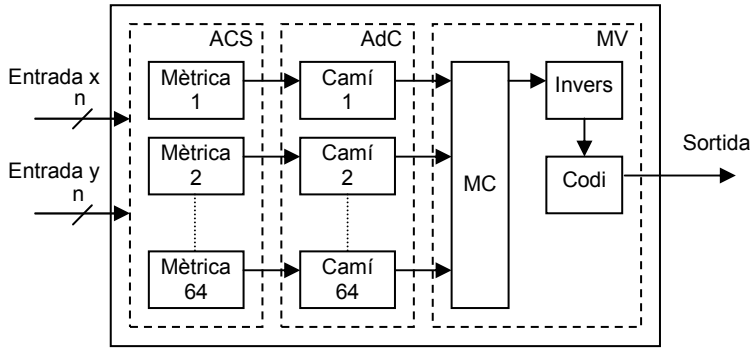


Figura 4.62 Diagrama de blocs del descodificador de 64 estats

b) Resultats

L'arquitectura utilitzada és de tipus paral·lel, per tant, es determina, en cada cicle de rellotge, la mètrica de cada branca i la mètrica acumulada associada a cada node. Llavors, tenint en compte que la mètrica es calcula mitjançant el producte escalar, es necessiten una gran quantitat de recursos de la FPGA, tal com mostra la Taula 4.29 (principalment, es concentren en el mòdul ACS). Per aquest motiu, s'han contemplat les següents alternatives per optimitzar recursos: la utilització de memòria distribuïda i l'eliminació de la redundància en el càlcul de la mètrica, ja que existeixen mòduls que realitzen la mateixa operativa amb las mateixes senyals d'entrada.

L'arquitectura sèrie s'ha desestimat ja que implica augmentar la velocitat de treball i seria impossible complir les restriccions temporals. Per exemple, pel cas de 54Mbits/s, si només es doblés la velocitat significaria unes restriccions mínimes de 108MHz.

FPGA Virtex II 2000	Viterbi
Slices	4.775
BRAMs	2
FFs	1.316
LUTs	8.877
Equivalent en portes	589.378

Taula 4.29 Recursos utilitzats per la implementació del descodificador convolucional

Cal tenir en compte que el descodificador de Viterbi és el bloc que té més complexitat computacional a la cadena HiperLAN/2.

c) Alternatives de disseny

Una alternativa de disseny és utilitzar el bloc de Viterbi de Xilinx. Aquest bloc permet la descodificació *Hard* o *Soft*. Per la descodificació *Soft* el bloc permet una entrada de 3 a 8 bits, la qual s'adapta a l'estàndard HiperLAN/2 que defineix una descodificació *Soft* de 4 bits amb 64 estats. El format de les dues entrades que utilitza una descodificació *Soft* de 4 bits, que accepta el bloc de Viterbi de Xilinx, es pot veure a la Taula 4.24. També aquest bloc inclou una entrada i sortida per controlar la validació de les dades.

Aquest bloc també s'ha utilitzat i ha permès la verificació de tota cadena conjunta de l'emissor i receptor. Els paràmetres d'entrada del bloc s'han fixat segons l'estàndard i els més importants són la K=7, el codi convolucional = [133 171], format = Magnitud sense signe, longitud d'exploració =35 i tipus d'arquitectura = paral·lel.

Cal destacar que el bloc de Viterbi de Xilinx adjuntat a la llibreria del System Generator només permet fer la simulació, ja que per la seva implementació es necessita llicència.

4.4.7. *Scrambler/Descrambler*

El bloc *scrambler* correspon a l'emissor però s'explica en aquest apartat per no repetir les mateixes explicacions, ja que aquest bloc és idèntic al *descrambler* del receptor.

Els bits de la seqüència PDU que arriben des de la capa DLC s'han de mesclar amb una seqüència pseudoaleatòria generada pel bloc *scrambler* a partir del polinomi X^7+X^4+1. L'objectiu d'aquest bloc és modificar el bits per tal d'evitar llargues seqüències de 0 o 1. La Figura 4.63 mostra el diagrama del bloc *scrambler* que és idèntic al *descrambler*.

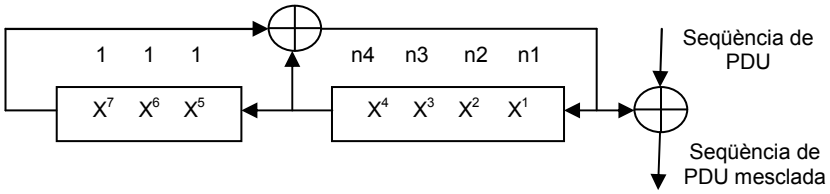


Figura 4.63 Diagrama esquemàtic del bloc *scrambler*

Cal tenir en compte que el bloc *scrambler* s'ha d'inicialitzar al valor (111n4n3n2n1) on n4,n3,n2 i n1 vénen donats per la capa MAC.

a) Disseny

El disseny d'aquests bloc és idèntic al presentat en el generador pseudoaleatori del bloc símbol OFDM amb la diferència de les inicialitzacions i d'una or-exclusiva de més, ja que el polinomi de partida és el mateix.

b) Resultats

La taula següent mostra els recursos utilitzats pels blocs *Scrambler* i *Descrambler*.

FPGA Virtex II 2000	*Scrambler/Descrambler*
Slices	9
Flip-Flops	7
LUTs	2

Taula 4.30 Recursos utilitzats per a la implementació dels blocs *Scrambler* i *Descrambler*

4.5. Resultats

S'han obtingut els models funcionals sintetitzables per unes restriccions de rellotge de 60MHz de l'emissor i receptor de l'HiperLAN/2 QPSK, amb una tassa de transmissió de dades de 12MBits/s. Cal comentar que la majoria dels blocs s'han fet parametritzables per facilitar i flexibilitzar futurs desenvolupaments de la resta de modes especificats a l'estàndard. A partir dels blocs dissenyats s'han pogut construir diversos models utilitzant algunes de les alternatives de disseny exposades en els apartats anteriors. Per exemple s'han fet dos models d'emissor i receptor HiperLAN/2 QPSK utilitzant el bloc IFFT/FFT dissenyat i un altre amb el bloc IFFT/FFT de Xilinx. També s'ha fet el mateix amb el bloc de Viterbi. Un exemple d'això, ha estat l'obtenció del model d'emissor i receptor 16QAM.

Tots els algorismes dissenyats s'han verificat amb la plataforma utilitzant la coverificació. Amb aquesta tècnica de verificació s'assegura la funcionalitat de l'algorisme sobre la plataforma. La figura següent mostra un exemple de coverificació de l'algorisme de l'equalitzador de canal. Es pot veure la cadena complerta del model de l'emissor i receptor HiperLAN/2 utilitzada. En aquest cas, l'equalitzador s'executa sobre la plataforma i interactua amb el model Simulink intercanviat dades d'entrada i sortida.

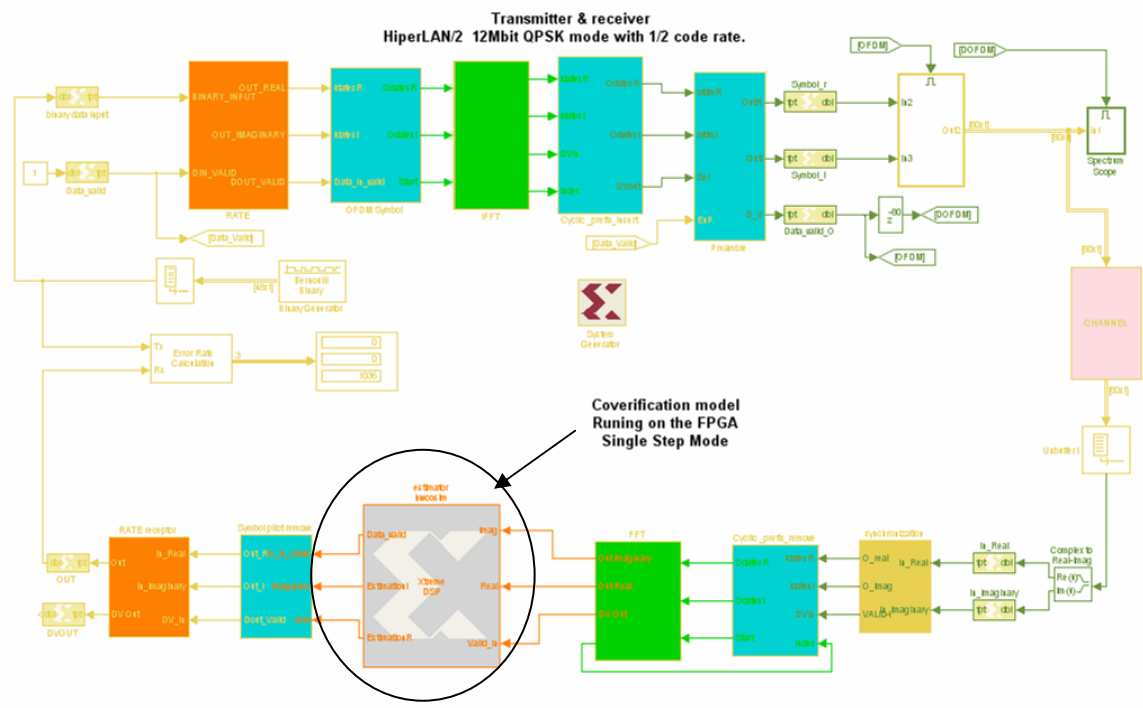


Figura 4.64 Exemple de coverificació

a) Recursos

A continuació es presenten els recursos utilitzats per l'emissor i receptor tenint en compte una FPGA Virtex II de Xilinx. Cal destacar d'aquests resultats, que el receptor necessita molts més recursos que l'emissor, ja que el receptor inclou la FFT i el descodificador de Viterbi, els quals són algorismes amb un elevada complexitat computacional. Per fer la síntesi de l'emissor i receptor s'ha fixat unes restriccions de rellotge de 16ns.

La Taula 4.31 mostra els recursos estimats parcials i totals dels blocs corresponents a l'emissor. Fent la síntesi de l'emissor utilitzant una FPGA Virtex II

2000 els recursos són de 2.268 *slices,* 12 BRAMs i 16 multiplicadors, equivalent a 950.263 portes.

Virtex II 2000	*Scrambler*	Codifi.	Inter.	Mapat	Símbol	IFFT	Capçalera	Total
Slices	9	3	28	13	152	1.118	845	2.107
BRAMs	0	0	1	0	1	10	0	12
Multiplicadors	0	0	0	0	0	16	0	16

Taula 4.31 Recursos estimats parcials i totals dels blocs de l'emissor

La Taula 4.32 mostra els recursos estimats parcials i totals dels blocs corresponents al receptor. Fent la síntesi total del receptor utilitzant una FPGA Virtex II 3000, els recursos són de 8.707 *slices*, 18 BRAMs i 27 multiplicadors, equivalent a 1.859.457 portes.

Virtex II 3000	*Descram.*	Viterbi	*Deinter.*	Desmapat	Extractor	Equalit.	FFT	Sincro.	Total
Slices	9	4.775	38	203	83	1.295	1.229	1.350	8.982
BRAMs	0	2	1	2	1	1	10	1	18
Multiplicadors	0	0	0	0	0	3	16	8	27

Taula 4.32 Recursos estimats parcials i totals dels blocs del receptor

Aquests resultats són acceptables i no són molt diferents als presentats en el llibre *OFDM Wireless Lans: A theoretical and Practical Guide* [3], on presenten els resultats d'un disseny d'un emissor i receptor d'OFDM IEEE 802.11a. Fent una valoració després de comparar resultats, tenint en compte les diferències de disseny, cal destacar que a nivell de portes equivalents els resultats són molt similars. També, un aspecte important és que nosaltres no hem utilitzat IP *cores* per implementar els algorismes més complexos.

b) Retards de l'emissor i del receptor

A la Taula 4.33 es mostren els retards del diferents blocs i el retard total de l'emissor. Cal tenir en compte que al retard de l'emissor s'hauria de descomptar el retard de la capçalera.

	Scrambler **Codificador** ***Interleaver*** **Mapat**	Símbol OFDM	IFFT	Retard Total
Retard (µS)	4,4167	4,0163	8,167	16.6

Taula 4.33 Retards de l'emissor

A la Taula 4.34 i a la Taula 4.35 es mostren els retards del diferents blocs i el retard total del receptor utilitzant el Viterbi desenvolupat a Vic i el Viterbi de Xilinx respectivament. Cal destacar que el Viterbi de Xilinx té un retard més gran. En aquesta taula no es contempla el retard del bloc de sincronisme.

	FFT	Equalitzador	Extractor de dades	Desmapat *Deinterleaver*	Viterbi *Descrambler*	Retard Total
Retard (µS)	8,5	9,313	4,27	4,334	6,25	32,667

Taula 4.34 Retards del receptor (Viterbi UVic)

	FFT	Equalitzador	Extractor de dades	Desmapat *Deinterleaver*	Viterbi *Descrambler*	Retard Total
Retard (µS)	8,5	9,313	4,27	4,334	13,416	39,983

Taula 4.35 Retards del receptor (Viterbi Xilinx)

c) Probabilitat de bits erronis (BER)

Per obtenir les gràfiques del bits erronis en funció del soroll, s'han generat ràfegues formades per seqüències de bits aleatòries. S'ha de considerar que per a cadascuna d'aquestes ràfegues el canal pot variar, per tant, és necessari tenir en compte aquest aspecte en les simulacions. Per aconseguir que el canal sigui diferent per a cadascuna

de les ràfegues, es canvia la llavor de forma aleatòria dels blocs de Simulink encarregats de generar el soroll AWGN i el canal multicamí. D'aquesta manera assegurem que el canal per a cadascuna de les ràfegues sigui diferent.

Cal recordar que per construir un símbol OFDM es necessiten més o menys bits en funció del mode d'HiperLAN/2 seleccionat. En aquest cas es tracta del mode 3 i necessita 48 bits per obtenir un símbol OFDM. Tenint en compte aquests paràmetres, s'han preparat ràfegues de 1008 bits que corresponen a 21 símbols OFDM. A continuació, s'ha fet una programa des de Matlab iteratiu amb Simulink (Figura 4.65) que controla la simulació i va acumulant les dades del bits erronis que es van produint. Per obtenir aquests resultats s'ha actuat de forma adaptativa per estalviar temps. Això vol dir que a mesura que s'augmenta la relació SNR el número de mostres estadístiques agafades s'incrementa, per tal d'assegurar uns resultats fiables.

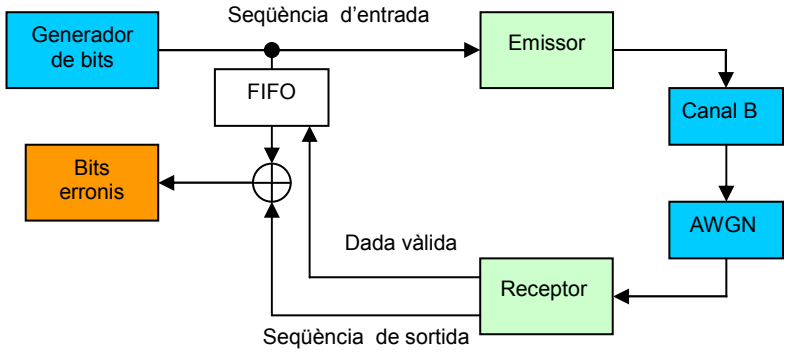


Figura 4.65 Diagrama de blocs per obtenir els BER en la cadena HiperLAN/2

La Figura 4.66 mostra el diagrama de blocs utilitzat per obtenir els bits erronis en la cadena de l'HiperLAN/2. Cal destacar que la memòria FIFO que apareix en la figura serveix per sincronitzar temporalment els bits d'entrada amb els de la sortida, per tal de poder fer una comparació de bits. Per veure clarament l'efecte del canal multicamí en la nostra cadena dissenyada, s'han obtingut dues gràfiques (Figura 4.66 i Figura 4.67) que mostren la probabilitat d'error BER en funció de la relació SNR, una utilitzant només un canal AWGN i l'altra afegint el canal B especificat per l'estàndard (canal Rayleigh amb un RMS delay spread de 100ns).

La figura següent mostra la probabilitat de bits erronis BER del nostre sistema HiperLAN/2 dissenyat en funció de la relació SNR per un canal AWGN.

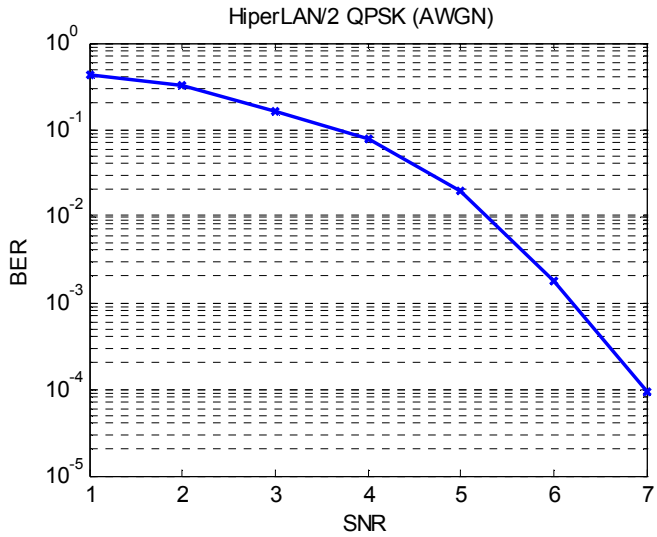


Figura 4.66 Probabilitat de bits erronis BER per un canal AWGN

En aquest cas, la figura següent mostra els BER del nostre sistema HiperLAN/2 dissenyat per el canal B especificat per l'estàndard.

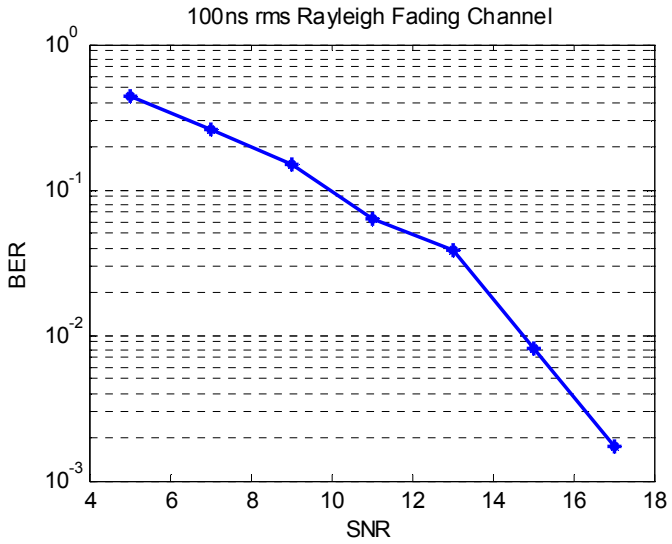


Figura 4.67 Probabilitat de bits erronis BER: canal Rayleigh amb un RMS delay Spread de 100ns (canal B de l'estàndard HiperLAN/2)

Analitzant aquestes gràfiques s'arriba a la conclusió que el nostre sistema funciona molt bé amb unes condicions de soroll o del canal que no siguin molt agressives. Cal considerar que s'han utilitzat algorismes de baixa complexitat, per exemple l'estimador de canal i la utilització d'una sola portadora pilot per fer el seguiment d'offset de freqüència. També, cal tenir en compte les simplificacions fetes en l'estimador de freqüència i de temps en el bloc de sincronisme.

4.6. Portabilitat tecnològica

Els models que s'han obtingut són descrits al nivell de RTL (*Register Transfer Level)* i es poden portar a diferents famílies de FPGAs de Xilinx. Un aspecte no massa conegut de System Generator i força interessant és la capacitat de crear RTL genèrics, els quals són molt eficients i fàcilment transportables a altres tecnologies [74]. Bàsicament hi ha tres camins per obtenir aquests RTL genèrics que són: important mòduls HDL a través de blocs *Black Boxes*, utilitzant blocs d'implementacions RTL bàsics, tals com registres, retards, ... i mitjançant el bloc Mcode que mapa fitxers Matlab.m a un codi VHDL sintetitzable. De totes maneres, la portabilitat de tecnologia quan s'utilitzen dispositius lògics programables (FPGAs) tot i que sembla directe si utilitzem descripcions de nivell lògic, depèn fortament de l'ús de recursos arquitecturals (memòries, elements de sincronització, processadors, IPs,...) en els dissenys, els quals ha estat optimitzats per a aquests tipus de dispositius i són més difícilment transportables tecnològicament.

Cal tenir en compte la transformació que ha experimentat l'ús de les FPGAs que han passat de ser dispositius utilitzats per a prototipus a ser utilitzats en producte final, ja que per a moltes aplicacions és possible tenir el producte abans al mercat amb la possibilitat d'actualització del hardware en cas de necessitat.

La portabilitat tecnològica FPGA/ASICs té un cost i un risc de funcionalitat i de restriccions temporals. De totes maneres, en el mercat existeixen solucions de conversió tecnològica per Xilinx i Altera [75], lògicament amb un cost, que asseguren la funcionalitat i les restriccions temporals.

També, els fabricants de FPGAs han cercat solucions intermitges, com per exemple, els dispositius HardCopy d'Altera [18] o la nova generació de FPGAs

EasyPath de Xilinx [17]. Els dispositius HardCopy són reproduccions d'un dispositiu lògic programable sense programabilitat amb una configuració específica i un connexionat implementat a través d'una interconnexió amb metall. Els dispositius EasyPath són FPGAs estàndards que han estat verificats per reunir els requeriments d'una aplicació específica. Per tant, pel cas del model obtingut, EasyPath de Xilinx ofereix una conversió lliure amb uns costos de producció inferiors a les tecnologies d'ASICs estructurats, tal com mostra la comparativa de costos entre diferents tecnologies de la Figura 4.68. Cal destacar que EasyPath garanteix la funcionalitat i les mateixes restriccions temporals que el prototipus.

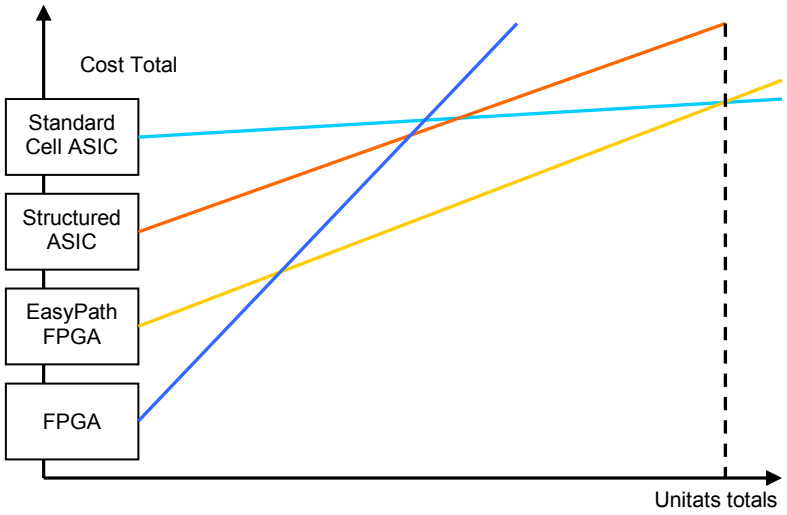


Figura 4.68 Gràfica comparativa de costos de producció entre diferents tecnologies

Per tant, s'arriba a la conclusió que aquesta tendència a la utilització de FPGAs com a producte final i a les noves tecnologies de conversió de dissenys realitzats amb FPGAs a dispositius intermitjos, donen un gran impuls a les metodologies de prototipatge ràpid per a FPGAs.

4.7. Sumari

En aquest capítol s'ha presentat el disseny bloc per bloc de l'emissor i receptor de l'estàndard HiperLAN/2, exposant els diferents conceptes teòrics necessaris, les tècniques de disseny utilitzades i les arquitectures de hardware derivades d'algorismes propis. Alguns dels blocs s'han fet parametritzables per tal d'adaptar-se als diferents modes de l'HiperLAN/2. El resultat obtingut és un model funcional sintetitzable per a les FPGAs de Xilinx de l'emissor/receptor QPSK amb una tassa de transmissió de 12 MBits/s. Aquest model es pot utilitzar en el futur per fer noves exploracions de hardware, així com la quantificació del senyals òptimes per tenir un hardware el màxim d'eficient. Cal destacar que la metodologia utilitzada permet la implementació d'algorismes complexos d'alta complexitat computacional des del nivell de sistema amb un resultats de recursos i temporals més que acceptables. La portabilitat tecnològica és un tema pendent pels fabricants de FPGAs que estan cercant solucions intermitges que siguin viables.

El model sintetitzable de l'emissor descrit en aquest capítol serveix de base per la implementació a una FPGA de Xilinx en el pròxim capítol.

Capítol 5. **Prototipatge de l'emissor**

En el capítol anterior s'ha obtingut el model funcional sintetitzable de la part digital de la capa física de l'emissor i del receptor, verificat a nivell de mòduls. L'objectiu d'aquest capítol és el prototipatge del model sintetitzable de l'emissor en una FPGA de Xilinx, amb la finalitat de demostrar la viabilitat de la metodologia de prototipatge ràpid. Per aconseguir aquest objectiu cal adaptar el model de l'emissor a la plataforma que es disposa.

Aquest capítol s'ha estructurat en cinc apartats. Al primer, es presenta la plataforma seleccionada per fer el prototipatge de l'emissor. Al segon, s'exposa les adaptacions que cal fer al model per prototipar-lo a la plataforma disponible. Al tercer, s'expliquen les estratègies de rellotge necessàries per la implementació del model. Al quart, s'exposen els diferents resultats que s'han obtingut de recursos, de consum i de visualització del senyal OFDM amb l'analitzador d'espectres. Al cinquè, es fa un sumari del capítol.

5.1. Plataforma Nallatech

A l'apartat 3.5 (metodologia de prototipatge d'un sistema WLAN) s'ha comentat que s'utilitzava la plataforma de Nallatech, la qual formava part del *kit* de desenvolupament XtremeDSP. És important abans de començar el prototipat de l'emissor conèixer les característiques principals d'aquesta plataforma.

La plataforma de Nallatech està formada per una placa mare (*BenONE*) i una placa filla (*BenADDA*), tal com mostra la Figura 5.1.

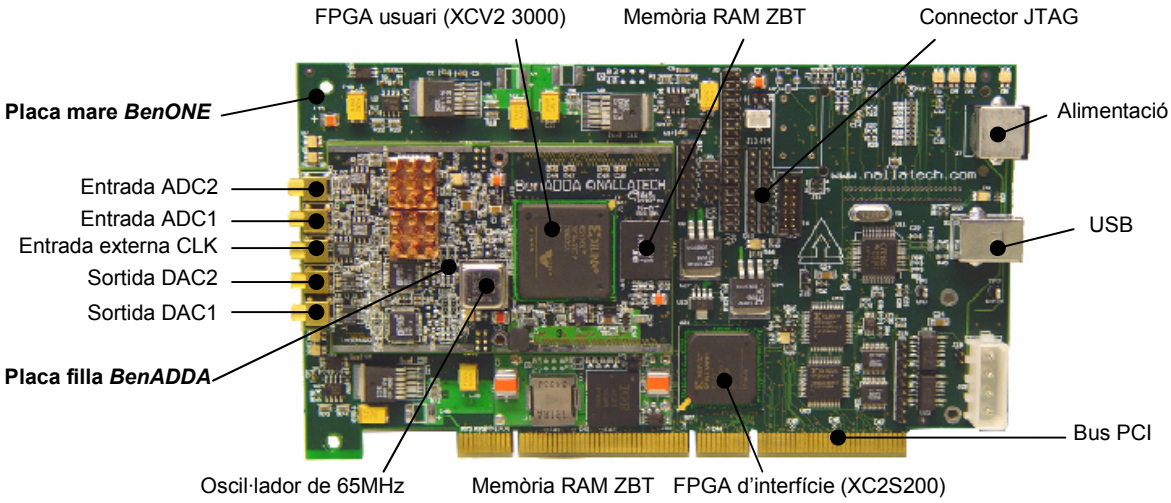


Figura 5.1 Plataforma Nallatech

Cal destacar les següents característiques hardware d'aquestes dues plaques:

- La placa mare *BenONE-Kit*
  - FPGA Spartan-II per 3.3V/5V amb interfície PCI o USB
  - Connector JTAG
- La placa filla BenADDA
  - FPGA Virtex-II XC2V3000-4FG676
  - 2 canals independents A/D: AD6644 ADC (14-bits fins a 65 Mmostres/s)

- 2 canals independents D/A: AD9772 DAC (14-bits fins a 160Mmostres/s)
- Una entrada externa de rellotge, un oscil·lador intern i rellotges programables
- Un banc de memòria ZBT-SRAM (133MHz, 512Kx16 bits)

A continuació, la Figura 5.2 mostra el diagrama de blocs de la plataforma de Nallatech.

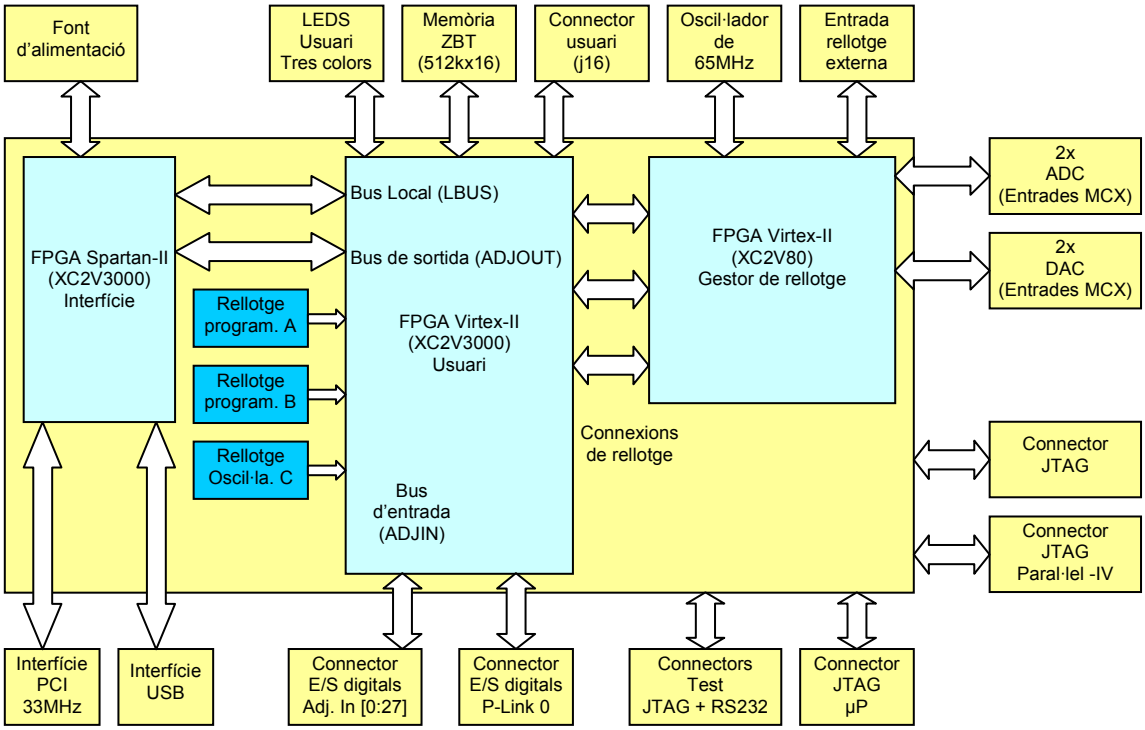


Figura 5.2 Diagrama de blocs de la plataforma de Nallatech

La Figura 5.3 mostra en més detall les connexions dels convertidors A/D i D/A, ja que aquests dispositius són essencials pel prototiptage del subsistema WLAN. Es pot veure que hi ha una FPGA petita (Virtex-II 2V80) que fa d'interfície entre els senyals de rellotge de la FPGA d'usuari i els convertidors A/D i D/A.

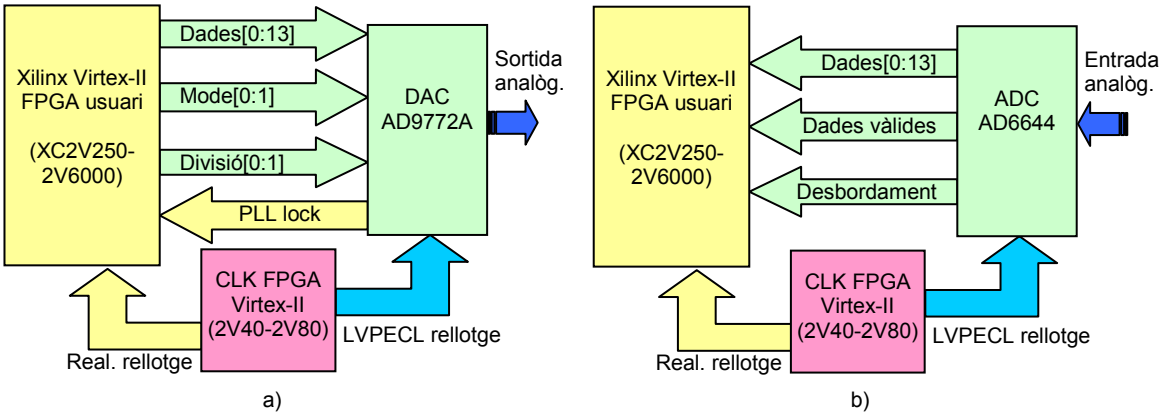


Figura 5.3 Detall de connexionat: a) convertidor DAC, b) convertidor ADC

Cal destacar del connexionat del convertidor D/A (Figura 5.3a) els senyals mode i divisió que serveixen per configurar-lo (resposta freqüencial del filtre reconstructor, rang d'entrada de les velocitats de mostreig de les dades, ...).

Un altre aspecte molt important d'aquesta plataforma són les possibles fonts i connexions dels rellotges de la plataforma. La Figura 5.4 mostra totes les possibles connexions entre les fonts de rellotge, FPGAs i convertidors A/D i D/A. Es pot veure en aquesta figura que la plataforma disposa de cinc fonts de rellotge, de les quals una és externa i la resta són internes. Per exemple la figura mostra una possible connexió utilitzant un oscil·lador programable com a font d'entrada de rellotge.

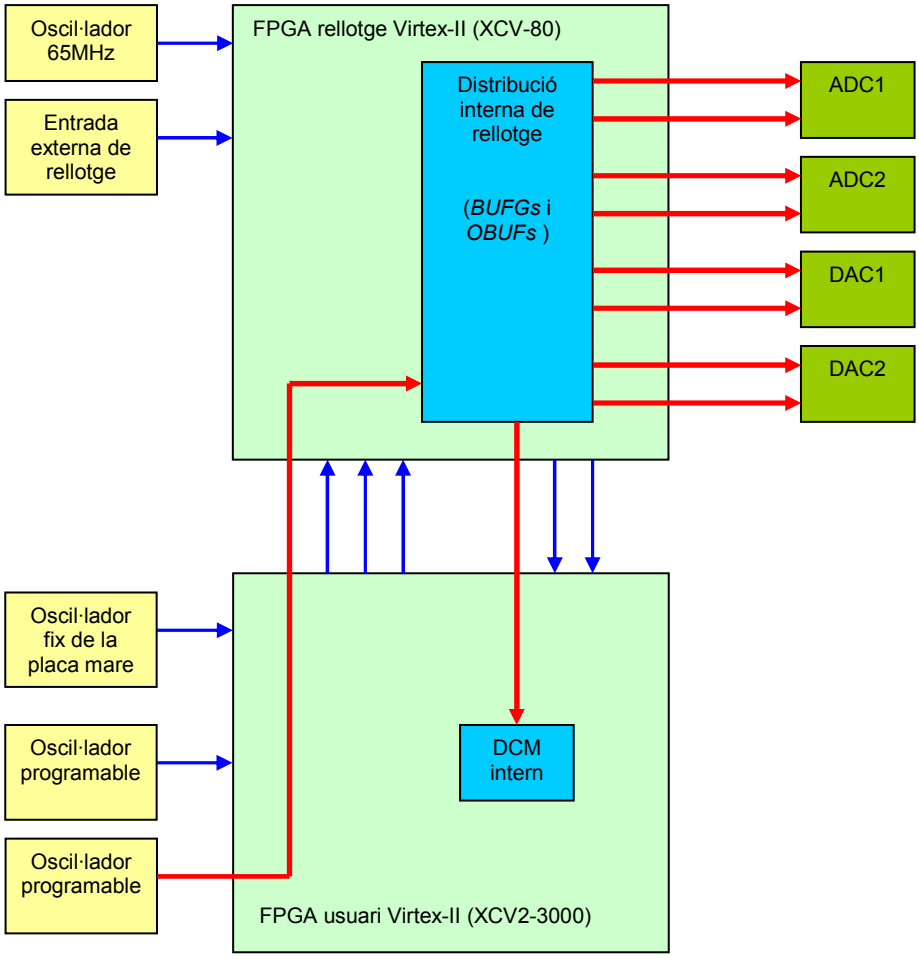


Figura 5.4 Fonts i connexions de rellotges per la FPGA d'usuari i els convertidors A/D i D/A

Després d'aquesta petita introducció a la plataforma Nallatech, s'ha pogut comprovar que aquesta s'adapta perfectament als sistemes DSP o WLAN. Cal destacar dos aspectes importants. El primer, la gran capacitat computacional de la FPGA d'usuari i les múltiples solucions de rellotge que ofereix. El segon, l'adaptació de la plataforma al flux de disseny presentat a l'apartat de metodologia. Per tant, creiem que és una de les plataformes més adequades pel prototipatge del nostre subsistema HiperLAN/2.

5.2. Model prototipat

L'adaptació que s'ha fet ha estat incloure els blocs interpolador, modulador i convertidor al final del model de l'emissor (Figura 5.5), per tal de poder veure l'espectre d'OFDM mitjançant un analitzador d'espectres. S'ha de tenir en compte que la capa física dissenyada és un model discret passabaix complex. Per tant, el senyal de sortida té dos components, una part real i una imaginària. Llavors, és necessari representar l'espectre real desplaçat a una freqüència intermitja per tal de visualitzar-lo

amb un analitzador d'espectres. També cal considerar que el senyal OFDM és discret en el temps, per tant, els seu espectre es repeteix en múltiples de la freqüència de mostreig. Això s'ha de tenir en compte per evitar que els espectres es sobreposin [71][72][73].

Per obtenir el senyal discret *y(n)* a la freqüència intermitja *f= wn/2π*, a fi de poder ser visualitzat en temps o freqüència cal construir-lo a partir de l'equivalent passabaixos, tal com mostra la següent expressió:

$$y(n) = \mathrm{Re}[r(n)] \cdot \cos(wn) - j\,\mathrm{Im}[r(n)] \cdot \sin(wn) \qquad (40)$$

On *Re[r(n))]* i *Im[r(n))]* són la part real i imaginària del senyal d'OFDM *r(n)* respectivament.

Cal decidir a quina freqüència intermitja es desplaça el senyal tenint en compte que la velocitat de mostreig del senyal d'OFDM de sortida és de 20Mmostres/s i que l'amplada de banda que fixa l'estàndard és de 20MHz. Cal considerar que la freqüència de mostreig és prou gran com per tenir suficients mostres per crear un sinus o cosinus a la freqüència intermitja amb una qualitat mínima. Un altre aspecte important, és que les dades s'han d'interpolar a la mateixa freqüència de mostreig que les funcions sinus i cosinus, per tal de poder realitzar els productes que apareixen a l'equació (40).

Per fer les operacions de l'equació (40) i la interpolació d'una forma òptima caldria una FPGA externa per poder treballar a més freqüència. En aquest cas, s'aprofita la mateixa FPGA, i s'ha decidit treballar a una freqüència de mostreig de sortida de 60Mmostres/s. El motiu d'aquesta freqüència de treball és que s'adapta a les necessitats de disseny de la capa física que es vol implementar i no perjudica en excés les restriccions temporals d'implementació. Cal destacar que aquesta decisió permet prototipar l'emissor d'OFDM utilitzant la placa de Nallatech, utilitzant el seu convertidor D/A. Per tant, per dur a terme les consideracions fetes fins ara, cal que s'afegeixin al model de l'emissor un bloc interpolador, un bloc modulador i un bloc DAC1.

El bloc interpolador de relació 1:3 s'implementa utilitzant el bloc *up sample* de Xilinx que permet augmentar el període de mostreig i un filtre FIR interpolador de 59 etapes. Per fer els càlculs dels coeficients s'ha utilitzat la funció de Matlab *intfilt*.

El bloc modulador consta de dos multiplicadors, un restador i el bloc DDS de Xilinx. Els multiplicadors són blocs dedicats de la Virtex II i el bloc DDS (*direct digital synthesizer*) és un sintetitzador digital directe. Aquest bloc utilitza una memòria per generar les funcions sinus i cosinus.

El bloc DAC1 és un bloc de la llibreria de Xilinx que permet configurar el convertidor DA de la placa Nallatech des de la FPGA. Els paràmetres que permet controlar són el *reset*, tipus de filtre (passa-baix o passa-alt) i rang de freqüència de mostreig de les dades d'entrada.

La següent figura mostra la situació d'aquest blocs a la cadena de l'emissor d'HiperLAN/2.

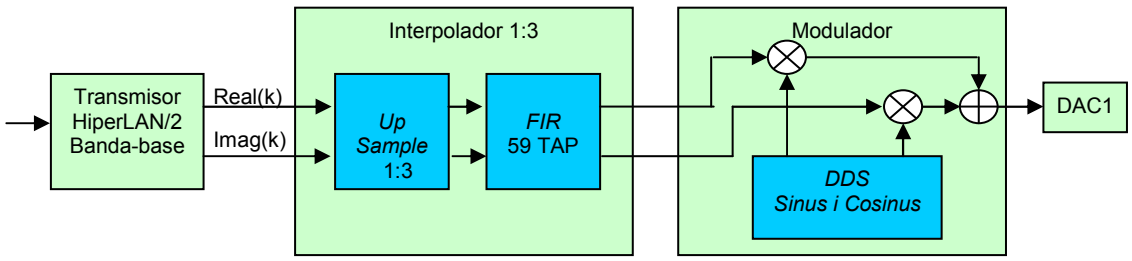


Figura 5.5 Blocs interpolador i modulador en la cadena de l'emissor HiperLAN/2

És important ubicar els espectres dels senyals discrets després de cadascun dels blocs de la Figura 5.6 per tal de centrar l'espectre d'OFDM en el lloc adequat, i també, per adaptar-lo correctament a les característiques del convertidor D/A que incorpora la plataforma de Nallatech.

La Figura 5.6 mostra l'espectre complex del senyal discret d'OFDM a la sortida de l'emissor hiperLAN/2 en banda base per a una freqüència de mostreig de 20Mmostres/s.

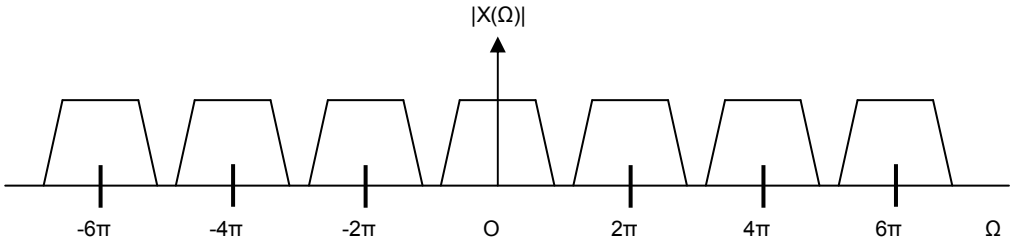


Figura 5.6 Espectre complex del senyal discret d'OFDM abans d'interpolar

La Figura 5.7 mostra l'espectre del mateix senyal discret a la freqüència de mostreig de 60Mmostres/s que s'obtindria després de passar pel bloc interpolador.

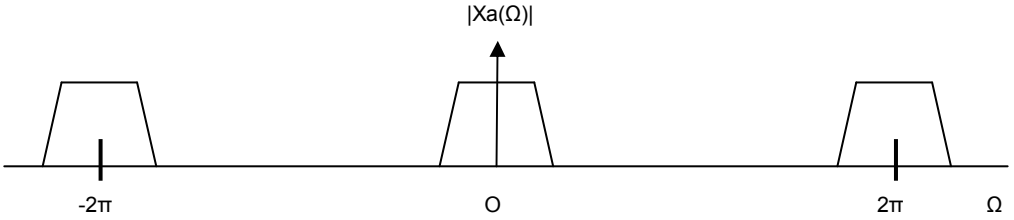


Figura 5.7 Espectre complex del senyal discret d'OFDM després d'interpolar

La Figura 5.8 mostra la distribució de les bandes de l'espectre del senyal discret d'OFDM després del bloc modulador. Cal tenir en compte que el fet de multiplicar per un sinus o un cosinus l'espectre es desplaça a la freqüència d'aquests però també es duplica l'espectre tal com mostra la figura. A la freqüència de mostreig de 60Mmostres/s, la freqüència del sinus i del cosinus hauria de ser tal que originés un desplaçament a les bandes de π/2, tal com es representa a la Figura 5.8. A la freqüència de mostreig de 60Mmostres/s la freqüència d'aquests hauria de ser de 15MHz.

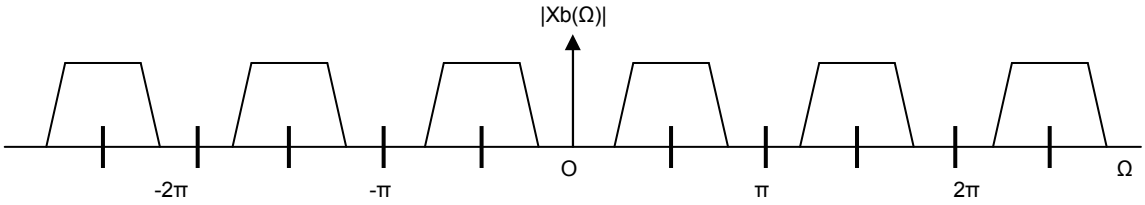


Figura 5.8 Espectre discret d'OFDM després del bloc modulador

El convertidor D/A que disposa la plataforma de Nallatech és un AD9772A de Analog Devices de 14bits que pot arribar fins a 160Mmostres/s amb un filtre interpolador 1:2. Per aplicacions en banda base el filtre interpolador es pot programar com a filtre passa-baix (Figura 5.9) i, per aplicacions directes de freqüències intermitges (IF) es pot programar com a passa-alt.

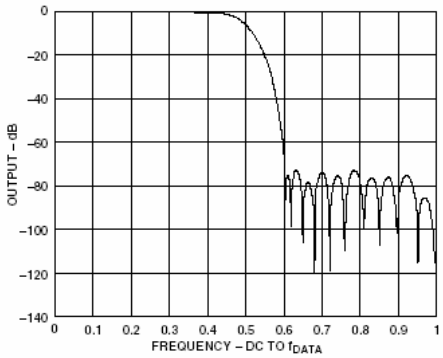


Figura 5.9 Resposta en freqüència del filtre interpolador FIR: passa-baix

La Figura 5.10 mostra el diagrama de blocs general del convertidor DA. Cal destacar que les dades són mostrejades a la freqüència a la qual arriben al convertidor (f_{dades}) i que el filtre interpolador i el DAC treballen al doble d'aquesta freqüència.

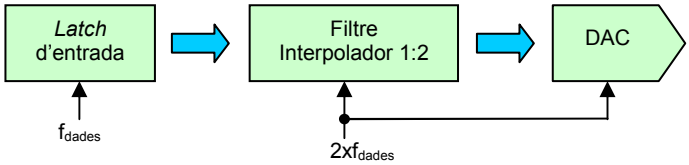


Figura 5.10 Diagrama de blocs general del DAC

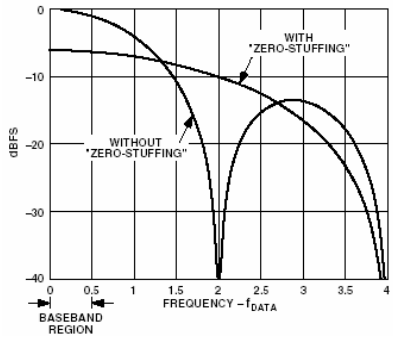


Figura 5.11 Resposta en freqüència del filtre reconstructor DAC

La Figura 5.11 mostra la resposta freqüencial del filtre reconstructor DAC sin(x)/x. Cal destacar les dues opcions programables de resposta del filtre reconstructor que ofereix el convertidor en funció si la aplicació és en banda base (sense *zero-stuffing*) o en IF (amb *zero-stuffing*).

5.3. Estratègia de rellotge

Un dels aspectes més importants a tenir en compte en el disseny de sistemes digitals és la gestió dels rellotges del sistema. El fet de treballar amb un entorn visual de flux de dades al nivell de sistema, com és l'entorn format per Simulink i System Generator, dóna uns mecanismes automàtics per a la gestió de rellotges de sistemes amb diferents tasses de transmissió de dades amb un sol domini de rellotge (veure apartat 3.5.6). El problema apareix quan la freqüència de rellotge del sistema assignada per Simulink no és implementable o el sistema té més d'un domini de rellotge. Simulink només pot treballar amb un sol domini de rellotge i assigna com a rellotge del sistema al mínim comú múltiple de tots els períodes que intervenen en el disseny. En aquestes condicions, seria interessant poder gestionar els rellotges des de l'entorn.

El System Generator proposa, per solucionar aquest problema, la partició del model en dos o més dominis de rellotges, ja siguin síncrons o asíncrons. L'objectiu d'aquesta partició és baixar a VHDL les diferents particions amb el rellotge corresponent. Un cop baixades a VHDL es tornen a pujar al sistema mitjançant *black box*. Aquesta maniobra fa que quan es genera el VHDL general del sistema des de Simulink, aquest no veu els períodes interns dels diferents *black boxs*. Després cal modificar el VHDL i si és necessari inserir un DCM (*Digital Clock Manager*) i fer les connexions oportunes.

L'emissor HiperLAN/2 és un exemple d'un sistema complex d'altes prestacions amb diferents tasses de transmissió de dades. En el cas de l'emissor QPSK que es vol implementar, Simulink assigna un rellotge de sistema de 120MHz, ja que el mínim comú múltiple de les freqüències que apareixen en el disseny (12,24,20 i 60MHz). Aquesta freqüència dispara les restriccions temporals d'implementació. Per tant, s'ha decidit partir el model en dos dominis: el domini A a 24MHz i el domini B a 60MHz. El punt de partició en la cadena de l'emissor s'ha fet en el bloc Símbol OFDM (Figura 5.12), ja que inclou una memòria RAM de doble port.

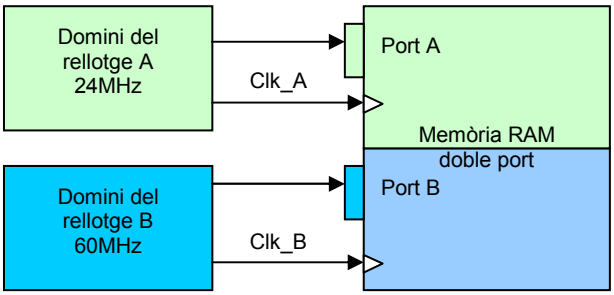


Figura 5.12 Partició del sistema en dos dominis de rellotge

Aquesta partició permetrà treballar amb un rellotge de sistema de 60MHz. Llavors els passos que cal seguir per la implementació de l'emissor són:

- Generar el VHDL del domini A amb un rellotge de 24MHz.
- Crear un nova jerarquia que inclogui el domini B i el domini A. Per incloure el domini A a la nova jerarquia es fa mitjançant un *black box* que conté el VHDL generat anteriorment. En cas de voler simular aquesta nova jerarquia cal recorre a la cosimulació, ja que existeix un bloc VHDL.
- Generar el Hardware de tot el sistema a través de System Generator. Automàticament en aquest procés es crea un bloc VHDL per cridar un DCM.
- Modificar el VHDL generat de tot el sistema per tal d'afegir al projecte de l'eina de síntesi (ISE) el DCM i per fer les modificacions oportunes per connectar les sortides del DCM als dominis de rellotge creats. També cal crear una connexió del DCM al convertidor DAC.
- Crear el fitxer de configuració de la FPGA mitjançant l'eina ISE de Xilinx.

Descarregar el fitxer de configuració de la FPGA a través de l'eina FUSE de Nallatech. La Figura 5.13 mostra l'estructura de connexió del DCM amb els dos dominis de rellotge de l'emissor i la connexió amb el convertidor DAC a través de la FPGA Virtex II XCV2-80. Aquesta FPGA gestiona els rellotges de tots els convertidors de la plataforma Nallatech. En aquest cas, converteix el senyal de rellotge provinent del boc DCM a una sortida diferencial de rellotge pel convertidor DAC. L'entrada de rellotge CLK s'aconsegueix mitjançant un oscil·lador programable que disposa la plataforma.

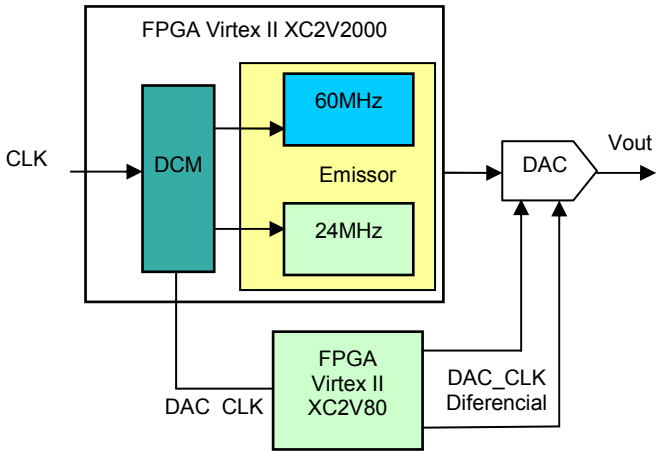


Figura 5.13 Estructura de connexió del DCM

Aquesta solució de gestió de rellotges permet la implementació de l'emissor de HiperLAN/2 d'una forma manual. Cal dir que aquesta opció no permet la descàrrega directa des de Simulink, ja que s'ha de fer des de l'eina de descàrrega de la plataforma Nallatech (FUSE). També, cal afegir que els fabricants de la plataforma han tret unes llibreries de Matlab que permeten la configuració de la plataforma des de Simulink i la descàrrega directe de fitxés de configuració de la FPGA de Xilinx.

5.4. Resultats

Recursos

La taula següent mostra els recursos utilitzats per implementar l'emissor de HiperLAN/2 en una FPGA Virtex II XC2V2000 de Xilinx.

FPGA Virtex II 2000	Memòria distribuïda
Slices	8.427
BRAMs	10
FFs	8.426
LUTs	6.721
Multiplicadors	13
DCM	1
Equivalent en portes	858.449

Taula 5.1 Recursos utilitzats per implementar l'emissor de HiperLAN/2

La Figura 5.14 mostra el dos dominis de rellotge des de l'entorn de Simulink.

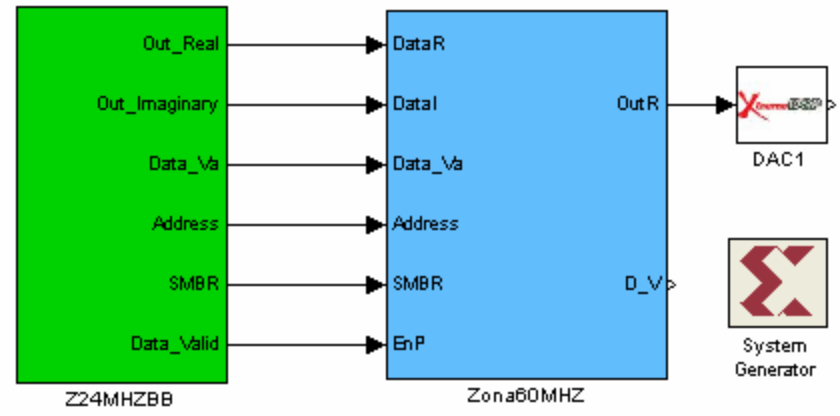


Figura 5.14 Emissor HiperLAN/2 prototipat

El domini de 24MHz inclou un bloc *black box* amb VHDL i el domini de 60MHz incorpora blocs de Xilinx. També es pot veure el bloc DAC1 que modela el control des de la FPGA del convertidor DAC. A partir d'aquest model es genera tot el VHDL de l'emissor.

Després de prototipar amb èxit l'emissor es fa una valoració molt positiva de la metodologia de prototipatge ràpid utilitzada on cal destacar:

- La capacitat de baixar i pujar blocs des del mateix entorn de Simulink. Això permet obtenir blocs VHDL, modificar-los i tornar-los a pujar al model.
- La modelació de blocs de la plataforma. Aquesta permet modelar els components a nivell de les necessitats del dissenyador. Per exemple, el cas del convertidor DAC només s'ha modelat la seva configuració, establint així una ortogonalitat entre el dispositiu i el model que facilita molt el canvi de plataforma.
- L'elecció adequada de la plataforma. Els resultats de síntesi de l'emissor es donen una ocupació d'un 43% de la FPGA XCV2000. Aquests resultats deixen espai suficient perquè en el futur s'implementi la interfície amb la capa MAC i la part de control de tota la capa física utilitzant IP (processador *soft*).

Verificació de l'emissor

Per comprovar el funcionament de l'emissor es connecta l'analitzador d'espectres a la plataforma, en concret a la sortida del DAC. La descàrrega de la configuració de la FPGA es fa a través del port USB del PC (Figura 5.15).

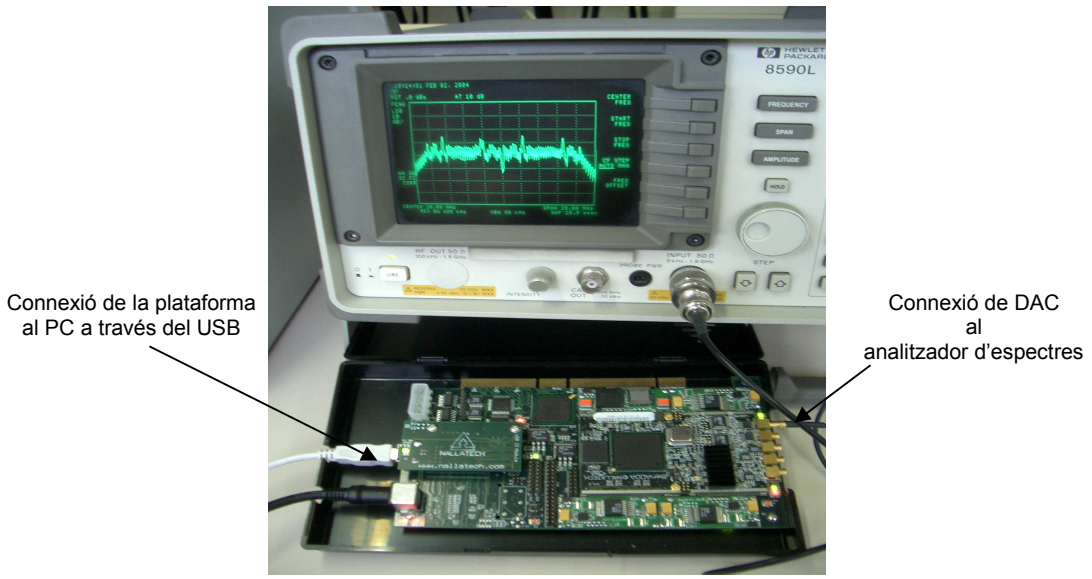


Figura 5.15 Connexió de la plataforma al analitzador d'espectres i al PC

Unes de les proves que s'han fet és anul·lar les dades i només deixar els símbols pilots, per tal de comprovar que sortien a la freqüència desitjada. Cal recordar que els símbols pilots s'han inserit a les subportadores -21,-7,7 i 21, per tant, la seva ubicació a l'espectre centrat a 15MHz són 8.44, 12,82, 17.2, 21.56MHz respectivament. La següent figura mostra l'espectre centrat a 15MHz amb una *span* de 20MHz. En concret, el cursor de l'analitzador d'espectres es troba situat en el pic de la subportadora 21, i el seu valor és de 21.60MHz.

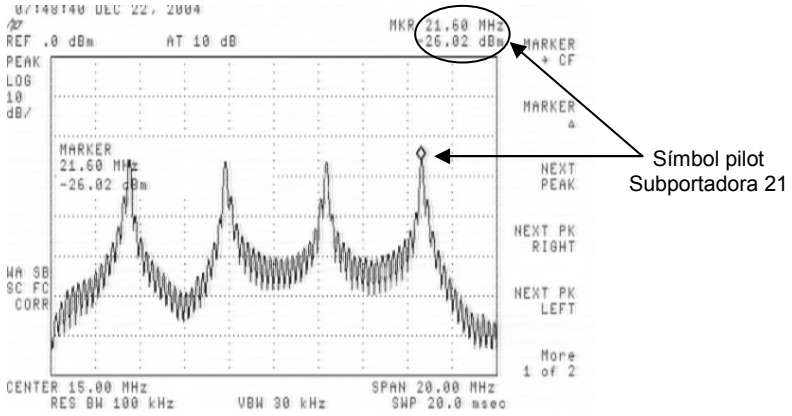


Figura 5.16 Espectre d'OFDM anul·lant les dades i només deixant els símbols pilot

A continuació s'han incorporat les dades però amb un 75% del seu valor en amplitud, modificant el valors dels símbols del bloc de mapat, per tal de veure l'espectre complert, els símbols pilots i la component contínua. La Figura 5.17 mostra clarament la situació dels símbols pilots que sobresurten respecte a les dades. També es pot apreciar perfectament la freqüència central de l'espectre, corresponent a la freqüència contínua de l'espectre passa baixos, la qual no porta informació.

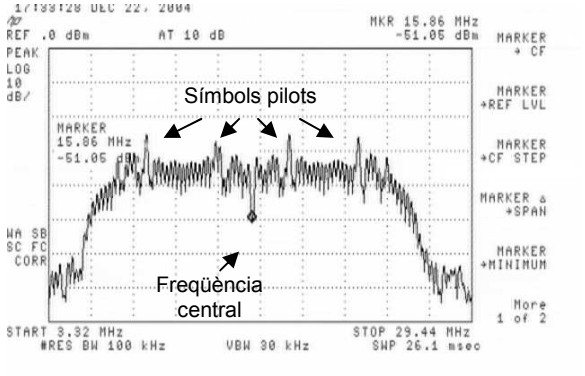


Figura 5.17 Espectre d'OFDM amb dades (75% del seu valor) i símbols pilots

Finalment s'ha comprovat l'espectre amb les dades i símbols. L'amplada de l'espectre mesurat és de 16.5MHz. Aquest valor s'ajusta al valor teòric de 16.25MHz.

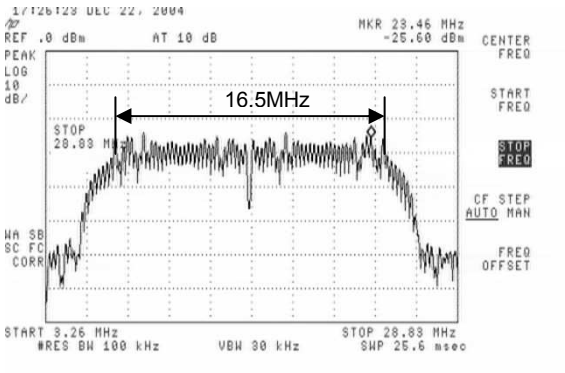


Figura 5.18 Espectre d'OFDM amb dades i símbols pilots

A través de l'analitzador d'espectres s'han cercat i es mostren els espectres mirall i múltiples més significatius.

La Figura 5.19 mostra els espectres real i imaginari del senyal d'OFDM centrats a 15MHz i a -15MHz respectivament.

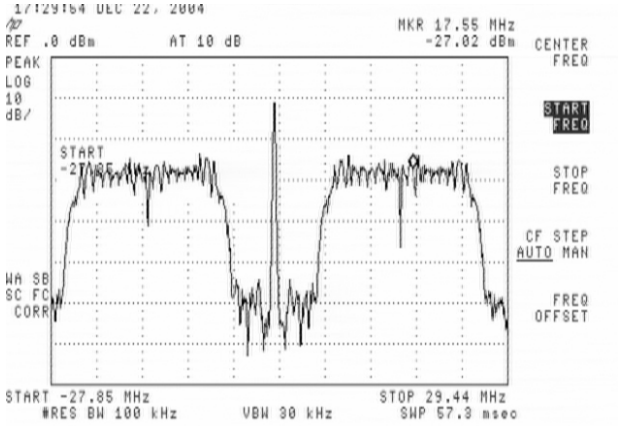


Figura 5.19 Espectres real i imaginari del senyal d'OFDM

La Figura 5.20 mostra els espectres principal i el seu primer múltiple. Cal destacar l'efecte dels filtres interpolador i reconstructor del DAC. El filtre interpolador del DAC desplaça l'espectre del primer múltiple al doble de la freqüència de mostreig i el filtre reconstructor atenua l'espectre múltiple del senyal OFDM. En concret, el primer zero del filtre reconstructor (Figura 5.11), cau exactament al doble de la freqüència de mostreig. Es pot apreciar clarament, a la figura següent, la forma de la *sinc* en l'atenuació de l'espectre múltiple.

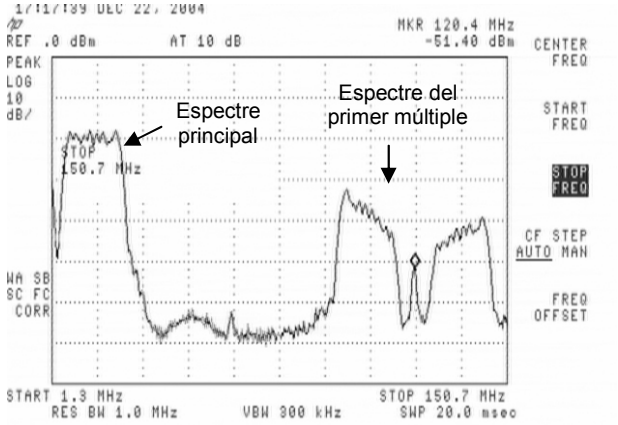


Figura 5.20 Espectres principal i el seu primer múltiple del senyal d'OFDM

Consum de potència

Per determinar la potència de l'emissor es poden fer mesures reals sobre la plataforma o utilitzar metodologies d'estimació de potència.

Per fer mesures reals del disseny es pot seguir la tècnica de fer la diferència entre les mesures de potència de tota la plataforma sense descarregar el disseny i descarregant el disseny. Aquesta tècnica dóna bons resultats si no intervenen dispositius externs, com per exemple convertidors DAC.

Una altra manera és fent estimacions de potència. En el cas de l'emissor que s'ha implementat en FPGAs de Xilinx, es pot fer l'estimació utilitzant l'eina XPower o també fent una estimació menys acurada aplicant els resultats d'una article de Xilinx sobre el consum de potència dinàmica [70]. Aquest article diu que per a dissenys típics utilitzant FPGAs Virtex II, el consum per CLB és de 5,9μW/MHz.

En aquest cas es fa l'estimació utilitzant l'eina XPower. El XPower calcula la potència basant-se en l'observació del consum dinàmic en circuits CMOS que bàsicament és degut a l'activitat de commutació. Llavors, l'exactitud de l'activitat de commutació és crucial per obtenir una bona estimació de consum. El XPower calcula el consum com una suma de la potència consumida per a cadascun dels elements *i* del disseny partint de la següent expressió:

$$P = \sum_i C_i \cdot V_i^{\,2} \cdot E_i \cdot f_i \cdot 1000 \qquad (41)$$

On P és la potència (mW), C_i és la Capacitat (fF), V_i és la tensió (V), E_i és l'activitat de commutació (mitja del número de transicions per cicle de rellotge) i f_i és la freqüència(Hz).

La capacitat C_i es determina a partir del disseny de l'usuari (per un disseny implementat en un dispositiu específic, fixat durant la caracterització dels recursos del connexionat i dels elements del dispositiu). La tensió V_i es fixa per un dispositiu específic. El XPower determina l'activitat de commutació dels elements en el disseny a través d'una de les següents possibilitats: fixant una activitat global per defecte, agafant l'activitat d'un fitxer VCD (*Value Change Dump*) generat mitjançant el simulador ModelSim i/o fixant l'activitat d'entrada de forma manual per l'usuari.

D'aquestes tres possibilitats s'ha escollit l'opció del fitxer VCD i els resultats obtinguts de l'estimació de potència de l'emissor després de la simulació es mostren a la següent taula:

Disseny **Fitxer de restriccions** **Fitxer VCD** **Dispositiu**	hiperlan2_xilinxbb_clk_wrapper hiperlan2_xilinxbb_clk_wrapper.pcf F:\Definitiu\hiperlan2_consum\transmisor.vcd 2v2000fg676-4	
	I(mA)	**P(mW)**
Potència total estimada		**1432**
Vccint 1.50V:	668	1002
Vccaux 3.30V:	100	330
Vcco33 3.30V:	30	100
Clocks:	96	144
Inputs:	1	1
Logic:	300	450
Outputs:		
Vcco33	28	94
Signals:	121	182
Quiescent Vccint 1.50V:	150	225
Quiescent Vccaux 3.30V:	100	330
Quiescent Vcco33 3.30V:	2	7

Taula 5.2 Resultats d'estimació de potència de l'emissor HiperLAN/2

Es pot veure a la capçalera de la Taula 5.2 els fitxers d'entrada que necessita l'eina XPower per fer l'estimació. Aquests són el fitxer de recursos del disseny, el fitxer de restriccions i el fitxer d'activitats VCD. Els resultats d'estimació de potència es mostren a la Taula 5.2 en tres grups. El primer grup mostra els resultats de la potència total estimada del disseny a través de les alimentacions de la lògica interna, de les entrades i de les sortides de la FPGA (*Vccint, Vccaux, Vcco33*). Al segon grup, es mostren de forma detallada els consums de potència dinàmica estimada dels rellotges, de la lògica interna i dels senyals interns. Al tercer grup, es mostrem els resultats de la potència estimada estàtica de les alimentacions de la lògica interna, de les entrades i de les sortides (*Quiescent- Vccint, Vccaux, Vcco33*).

Segons aquests resultats la potència estimada del *core* del nostre disseny és de 1,002W. Llavors, amb aquest resultat i el número de portes equivalents (858.449

portes) del disseny presentat a la Taula 5.1, es calcula el consum estimat per porta (W/gate) que és de 1,167mW/gate. Considerant que les portes equivalents són portes NAND de dues entrades, es podria trobar una relació de consum i àrea per la implementació de l'emissor en un ASIC.

A continuació es presenta una taula on es mostren els nostres resultats de potència estimada i els resultats de potència dels ASICs d'OFDM IMEC [76], Radiata [77] i Hipersonic [78]. Aquesta taula no pretén ser una comparativa, ja que no sabem amb exactitud quina part de la cadena del transmissor HiperLAN/2 inclouen els ASICs ni tampoc el mode de transmissió utilitzat per fer les mesures de potència.

	Disseny prototipat FPGA XCV2 2000		**IMEC ASIC CMOS 0,18µm**		**Radiata CMOS 0,25µm**	**Hipersonic CMOS 0,18µm**
	1,5V core	3,3V I/O	1,8V core	3,3V I/O		1,8V core i 3,3V I/O
Potència	1,002W	428mW	43mW	156mW	290mW	300mW a 800mW

Taula 5.3 Consum de potència de l'emissor prototipat i de diversos ASICs d'OFDM

Com és lògic es pot veure en aquesta taula que la potència consumida pel nostre disseny implementat en una FPGA és més gran que la potència consumida per l'ASIC. Aquesta diferència de consum de potència és degut a que la FPGA necessita més transistors per configurar les LUTs i per programar el connexionat. Llavors per un disseny típic amb FPGAs, bastants d'aquests transistors no s'utilitzen però si que tenen un consum mínim (*leakage current*). Aquesta relació de consum de potència entre les FPGAs i els ASICs pot variar molt en funció de les tecnologies que es vulguin comparar. A la literatura en general parlen d'un factor de relació aproximat de 10 a 25 [9][79]. Considerant aquesta relació, els resultats de consum de potència del nostre disseny s'aproximen força als resultats dels ASICs mostrats a la taula.

5.5. Sumari

En aquest capítol s'ha prototipat el model funcional sintetitzable de l'emissor obtingut en el capítol anterior en la plataforma de Nallatech, validant la metodologia de prototipatge des de Matlab/Simulink a plataforma. S'han exposat el canvis necessaris en el model per la seva adaptació a la plataforma de Nallatech. Cal destacar les estratègies de rellotge implementades per tal de fer possible la implementació del model. A nivell de resultats s'han presentat els valors d'ocupació de l'emissor en la FPGA, s'han fet mesures amb l'analitzador d'espectres i s'han fet estimacions de potència del model per tal de tenir futures referències per possibles comparacions en implementacions ASICs.

En treballs futurs en preveu el prototipat del model funcional sintetitzable del receptor per tal de fer les mesures de tot el sistema implementat.

Capítol 6. **Conclusions**

En aquest capítol es presenten les compilacions dels resultats i les conclusions sobre la metodologia proposada, les arquitectures explorades i els models obtinguts. En l'etapa de realització d'aquesta tesi s'han anat obtenim resultats en diferents apartats, els quals s'han presentat en forma de presentacions i publicacions en diversos congressos i workshops nacionals i internacional [20][21][38] [47][48][49][50][59][63][64][67][71][72][73]. També cal considerar que la realització d'aquesta tesi comporta una continuació en unes línies de treball futur.

6.1. Introducció

La capa física de l'estàndard l'HiperLAN/2, és un subsistema complex d'altes prestacions amb algorismes amb requeriments computacionals elevats i amb unes restriccions temporals exigents, en tant que que gestionen tasses de transmissió de dades diferents que dificulten la sincronització de les estructures computacionals.

Per una altra banda, els fabricants de sistemes de comunicació utilitzen cada vegada més els dispositius lògics programables (FPGAs) en els seus dissenys. La flexibilitat que ofereixen aquests dispositius en la reconfigurabilitat del hardware, en la implementació d'arquitectures amb un alt nivell de paral•lelisme i les noves solucions tecnològiques intermitges que ofereixen els fabricats entre ASICs i FPGAs, els fan molt atractius per ser utilitzats com a producte final. A més, la tendència d'aquests dispositius, apart d'anar incrementant la seva capacitat en portes, és la incorporació de cores de processadors i de hardware dedicat, que els fan ideals per prototipar sistemes DSP i sistemes SoC.

La contribució metodològica essencial d'aquesta tesi ha estat proposar i adaptar una metodologia de disseny a nivell de sistema que ha permès l'exploració i implementació de noves arquitectures hardware d'alguns dels algorismes més complexos de la cadena HiperLAN/2. Els resultats que s'han obtingut són: els models sintetitzables per a FPGAs de la part digital de la capa física d'un emissor i d'un receptor HiperLAN/2, que han permès validar sobre plataformes físiques la metodologia i les arquitectures hardware dissenyades; el prototipatge de la part digital de la capa física de l'emissor HiperLAN/2 amb una tassa de transmissió de dades de 12Mbits/s en una plataforma específica per a sistemes DSP o WLAN, que ha permès validar la cadena de disseny que porta des de l'especificació en Matlab fins la placa de prototipatge, amb l'ajut de la verificació multinivell, establert per la metodologia.

S'ha fet una aportació des del disseny de sistemes HW/SW a la metodologia, al prototipatge i a les arquitectures per als sistemes de comunicacions DSP o WLAN, que obre nous camins de recerca i ajuda a desenvolupar i validar sistemes de comunicació d'aplicació específica.

6.2. Metodologia

Abans de començar, cal fer referència als orígens d'aquesta línia de recerca, cristal·litzats en el treball experimental de doctorat [20] on es va presentar la personalització a una plataforma de la metodologia de codisseny Hardware/Software per a sistemes reactius, basada en POLIS, la qual es va aportar experiència en metodologies de codisseny, la conceptualització del concepte de plataforma així com diverses publicacions en aquest camp [21][22][23].

La metodologia presentada en aquesta tesi està en canvi orientada als sistemes de flux de dades, però es pot estendre fàcilment a altres models i aplicacions que aprofitin les tecnologies de FPGAs i treguin profit de l'ús d'eines similars des de l'entorn Maltlab/Simulink [18] (p.e. processament d'imatges, ...). El desenvolupament dels diversos blocs més complexos de la cadena HiperLAN/2 durant la tesi, han aportat diverses publicacions [38][47][48][49][50][59][63][67] que han pogut ser desenvolupats i validats de forma eficient gràcies a la metodologia de prototipatge ràpid utilitzada.

S'ha adaptat i validat una metodologia de prototipatge ràpid, que ofereix un camí directe de disseny des d'un entorn a nivell de sistema a una plataforma de prototipatge amb FPGAs (Nallatech). Les eines Matlab/Simulink, System Generator són importants dins d'aquesta metodologia i formen un entorn visual molt potent de disseny de sistemes electrònics per a FPGAs de Xilinx. L'entorn Matlab/Simulink és una plataforma excel·lent per a l'exploració i validació d'algorismes. A més, disposa d'un entorn gràfic amigable i de llibreries que inclouen molts algorismes, entre les que cal destacar les de comunicació i de processament del senyal. L'eina System Generator disposa d'un gran ventall de blocs descrits a alt nivell d'abstracció, que van des de un simple registre fins a blocs complexos de comunicació. La possibilitat de combinar en el disseny llibreries del Matlab/Simulink i del System Generator, conjuntament amb els generadors d'estímuls i visualitzadors, fan que aquest entorn sigui adequat pel disseny de sistemes DSP. Dins la metodologia de disseny s'han presentat dos fluxos de disseny que utilitzen de forma intensiva la verificació heterogènia multinivell.

Cal destacar que aquesta metodologia permet dissenyar algorismes complexos a nivell de sistema amb una productivitat i eficiència elevada, ja que permet explorar-los en diferents nivells d'abstracció d'una forma òptima.

6.3. Arquitectures explorades

El desenvolupament de les diferents arquitectures explorades en la tesi, s'ha acompanyat sempre d'una justificació teòrica o matemàtica, per tal de consolidar-ne els resultats.

S'han explorat noves arquitectures hardware d'alguns dels blocs de la cadena HiperLAN/2, les quals ens han aportat innovacions fruit de la investigació publicades com a tals en diferents congressos. Cal destacar-ne els blocs de la IFFT/FFT [47][48][49][50], de l'equalitzador de canal [63][64], del sincronisme [59], d'*Interleaver/Deinterleaver* [71] i de Viterbi [67].

El bloc de la IFFT/FFT amb un grau de paral·lelisme, se li ha donat una flexibilitat addicional a partir de dissenyar una estructura regular de connexions entre les diferents etapes, diferent de la tradicional papallona. S'ha derivat un algorisme de radix 2 que canvia els gràfics de flux de dades que descriu el clàssic algorisme de Cooley-Tukey [45] i s'ha readaptat amb l'objectiu d'obtenir un nou ordre, partint de les propietats de delmació en temps i en freqüència de la transformada discreta de Fourier. Aquest s'ha adaptat a la restriccions temporals de l'estàndard HiperLAN/2, però també és fàcilment adaptable a altres aplicacions que necessitin paral·lelisme amb restriccions temporals exigents.

Al bloc equalitzador s'ha proposat un nou mètode d'estimació del quocient Y/X, el qual estima el canal utilitzant els símbols C de les capçaleres de HiplerLAN/2. Aquest mètode ofereix una computació ràpida i poc complexa. S'han utilitzat el mètode de Hung [61] per fer l'invers, ja que aquest algorisme no necessita iterar i l'algorisme de CORDIC [60] per fer les rotacions i els càlculs dels angles de desfasament. El resultat obtingut és un equalitzador de canal simple, robust amb una ocupació de recursos mínima.

El bloc de sincronisme s'ha dissenyat a partir de les simplificacions de l'algorisme clàssic de sincronisme en freqüència i en temps d'en Schmidl-Cox [53] proposades per en Hanzo-Keller [54]. Aquestes simplificacions han permès obtenir un bloc de sincronisme fàcilment implementable amb uns resultats de recursos acceptables.

El bloc d'*Interleaver /Deinterleaver* utilitza una memòria RAM de doble port que permet ordenar i desordenar a nivell de bit o de paraula. Cal destacar que el bloc dissenyat permet implementar l'algorisme de l'*Interleaver* i l'algorisme del *Deinterleaver*, només canviant els mòduls dels comptadors que fan la lectura i escriptura de la memòria.

El bloc de Viterbi dissenyat a la Universitat de Vic amb la mateixa metodologia, s'ha adaptat a la cadena de l'HiperLAN/2. Cal esmentar que aquest bloc utilitza una arquitectura paral·lela basada en el diagrama de Trellis [67][68][69].

6.4. Models HiperLAN/2

S'han obtingut models funcionals sintetitzables a nivell de sistema de la part digital de la capa física de l'emissor i del receptor de l'HiperLAN/2, amb una tassa de transmissió de dades de 12MBits/s. Aquests models han servit per validar la cadena sencera de la capa física de l'HiperLAN/2 i explorar les diferents arquitectures dels algorismes derivats. També s'han aportat tècniques de sincronització de dades i de gestió dels rellotges per a aquests tipus de sistemes amb diferents tasses de transmissió de dades. Cal destacar d'aquests models que són fàcilment parametritzables gràcies a la interacció entre els blocs de Xilinx i els blocs de Simulink.

A partir del model sintetitzable de l'emissor, s'ha prototipat el model de l'emissor HiperLAN/2 a la plataforma Nallatech, validant el camí directe des de Matlab/Simulink a plataforma [71][72][73]. Per fer aquest prototipat ha estat necessari afegir els blocs interpolador i modulador per obtenir el senyal discret d'OFDM a una freqüència intermitja, a fi de poder visualitzar en temps o freqüència. Llavors, a través de la capacitat de baixar i pujar blocs des del mateix entorn de Simulink s'ha fet la gestió de dos dominis de rellotge síncrons que ha fet possible el prototipatge del model emissor. Un altre aspecte important és el modelatge de components de la placa a nivell de sistema, ja que ha permès en el nostre model la configuració del control del convertidor D/A mitjançant la FPGA.

Fer una comparació amb altres sistemes sempre es complicat, de totes maneres cal considerar que sistemes semblats (IEEE802.11a) [3] s'han prototipat amb metodologies similars, però utilitzant sempre *cores* pels blocs més complexos (IFFT, FFT i Viterbi). En el cas d'aquesta tesi, aquests blocs s'han dissenyat des del nivell de sistema amb uns resultats molt bons (recursos d'ocupació de la FPGA i temporals). Els resultats d'estimació de potència ens han donat una primera aproximació del consum del nostre emissor en comparació a solucions ASICs.

6.5. Línies de treball futur

Les línies de treball futur poden ser moltes, ja que els sistemes de comunicació basats en OFDM són per tot arreu. De totes maneres, com a treballs futurs derivats d'aquesta tesi a curt termini, s'haurien de centrar bàsicament en completar la capa física de l'HiperLAN/2 i implementar la interfície amb la capa MAC. El fet d'implementar la interfície de la capa MAC, suposa treballar en mòduls software. Això implica un canvi en la metodologia de prototipatge, que s'hauria d'encaminar més capa metodologies de codisseny HW/SW. Una línia de recerca actual paral·lela aquesta tesi, és la computació reconfigurable de sistemes WLAN en sistemes SoC. Alguns dels algorismes que surten en la cadena de la capa física de l'HiperLAN/2, només actuen

durant un temps determinat de la ràfega (sincronisme, estimació de canal). Llavors, la idea, és que la resta de temps es pot reconfigurar el HW per a altres necessitats [9]. També cal considerar, pensant a mig termini, el desenvolupar un demostrador complet d'HiperLAN/2, el qual pot obrir aportat noves línies de recerca, sobretot per desenvolupar sistemes de comunicació propis.

Apèndix A

Delmació en temps

Considerant la matriu unitària de Fourier $F_{(n)}$ de dimensions *nxn* i formulant l'algorisme conegut de delmació en temps amb notació matricial queda:

$$F_{(n)} = \frac{1}{\sqrt{2}} \begin{bmatrix} I_{(n/2)} & D_{(n/2)} \\ I_{(n/2)} & -D_{(n/2)} \end{bmatrix} \cdot \begin{bmatrix} F_{(n/2)} & \\ & F_{(n/2)} \end{bmatrix} \cdot [P_{(n)}] \tag{42}$$

On $F_{(n/2)}$ és la matriu unitària *n/2xn/2*, $P_{(n)}$ és la matriu de permutació de files parell-senar, on cadascuna de les files de la matriu està formada per elements $P_{i,j}$ que valen tots zeros excepte un i la posició d'aquest element és:

$P_{i,j} = 1$ per *j = parell-senar (i)* ; *i = 1...n*,

$I_{(n)}$ és la matriu identitat de capacitat *nxn* i $D_{(n)}$ és la matriu diagonal i es defineix com:

$D_{(n)} = Diag\left\{e^{-j\pi k/n}\right\}$ per *K = 0,1,...,n*

Iterant l'equació (42) per *n/2* i així successivament fins a reduir la submatriu de Fourier $F_{(n/2)}$ a $F_{(2)}$, un total de $log_2(n)$ iteracions. Aquest procés es pot veure a la següent expressió:

$$F_{(n)} = \frac{1}{\sqrt{2}} \begin{bmatrix} I_{(n/2)} & D_{(n/2)} \\ I_{(n/2)} & -D_{(n/2)} \end{bmatrix} \cdot \frac{1}{\sqrt{2}} \cdot \begin{bmatrix} I_{(n/4)} & D_{(n/4)} & & \\ I_{(n/4)} & -D_{(n/4)} & & \\ & & I_{(n/4)} & D_{(n/4)} \\ & & I_{(n/4)} & -D_{(n/4)} \end{bmatrix} \cdots \begin{bmatrix} F_{(2)} & & & \\ & \ddots & & \\ & & & F_{(2)} \end{bmatrix} \cdot [BR_{(n)}] \tag{43}$$

Per a cada nova iteració apareix una nova matriu de permutació. La matriu producte de totes aquestes matrius és la matriu de reordenació *bit-reversed* $BR_{(n)}$.

Si s'observa la forma de les matrius i considerant que F_2 és igual a $B_{i(n)}$ per $i=(log_2 n)-1$, s'arriba a una expressió més compacte:

$$F_{(n)} = \frac{1}{\sqrt{n}} \cdot \left(\prod_{i=0}^{(\log_2 n)-1} Diag\{B_{(n/2^i)}\} \right) \cdot [BR_{(n)}] \tag{44}$$

On l'operador *Diag{ }* construeix una matriu diagonal per blocs a partir dels subblocs $B_{(n/2)}^{i}$. A cadascun d'aquests multiplicadors se'ls denomina etapes. La definició general d'aquests subblocs és:

$$B_{(n/2^i)} = \begin{bmatrix} I_{(n/2^{i+1})} & D_{(n/2^{i+1})} \\ I_{(n/2^{i+1})} & -D_{(n/2^{i+1})} \end{bmatrix} \tag{45}$$

Per aconseguir etapes amb una mateixa arquitectura, s'insereix entre les dues primeres etapes una matriu de permutació parell-senar $P_{(n)}$ multiplicada per la seva inversa $P_{(n)}^{-1}$ (per anul·lar el seu efecte), entre la segona i la tercera dues matrius de permutació $P_{(n)}^{2}$ amb les seves respectives inverses $P_{(n)}^{-2}$, entre la tercera i la quarta tres matrius de permutació $P_{(n)}^{3}$ amb les seves respectives inverses $P_{(n)}^{-3}$, ..., i així fins a completar totes les etapes. L'expressió que s'obté és la següent:

$$F_{(n)} = \frac{1}{\sqrt{2}}\begin{bmatrix} I_{(n/2)} & D_{(n/2)} \\ I_{(n/2)} & -D_{(n/2)} \end{bmatrix}.\left(\left[P_{(n)}\right]^{\log_2 n-1}\right)^{-1}.\left[P_{(n)}\right]^{\log_2 n-1}.$$

$$\frac{1}{\sqrt{2}}.\begin{bmatrix} I_{(n/4)} & D_{(n/4)} & & \\ I_{(n/4)} & -D_{(n/4)} & & \\ & & I_{(n/4)} & D_{(n/4)} \\ & & I_{(n/4)} & -D_{(n/4)} \end{bmatrix}.\left(\left[P_{(n)}\right]^{\log_2 n-2}\right)^{-1}.\left[P_{(n)}\right]^{\log_2 n-2}\ldots$$

$$\left(\left[P_{(n)}\right]\right)^{-1}.\left[P_{(n)}\right]\frac{1}{\sqrt{2}}.\begin{bmatrix} F_{(2)} & & \\ & \ddots & \\ & & F_{(2)} \end{bmatrix}.\left[BR_{(n)}\right] \tag{46}$$

Les matrius introduïdes entre les etapes no incrementen la complexitat matemàtica de l'algorisme, ja que aquesta està associada al número de multiplicacions. Les matrius de permutació només aporten una modificació dels pesos de cada etapa que permet canviar la connexió entre etapes. Les etapes són modificades perquè inclouen, en l'etapa *i*, premultiplicacions amb la matriu de permutació $P_{(n)}^{i-1}$ en el costat dret i postmultiplicacions amb $P_{(n)}^{-i}$ en el costat esquerra.

Aquest procés es veu més clar amb un exemple de dimensions reduïdes (*n*=*8*).

$$F_{(8)} = \frac{1}{\sqrt{2}}\underbrace{\begin{bmatrix} 1 & & & & 1 & & & \\ & 1 & & & & e^{-j\frac{2\pi}{8}} & & \\ & & 1 & & & & -1 & \\ & & & 1 & & & & e^{-j\frac{6\pi}{8}} \\ 1 & & & & -1 & & & \\ & 1 & & & & -e^{-j\frac{2\pi}{8}} & & \\ & & 1 & & & & 1 & \\ & & & 1 & & & & -e^{-j\frac{6\pi}{8}} \end{bmatrix}}_{B_2}.\underbrace{\begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{bmatrix}}_{\left(\left[P_{(8)}\right]^2\right)^{-1}}.\underbrace{\begin{bmatrix} 1 & & & & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & & & & 1 \end{bmatrix}}_{\left[P_{(8)}\right]^2}.$$

$$\frac{1}{\sqrt{2}}\underbrace{\begin{bmatrix} 1 & 1 & & & & & & \\ 1 & -1 & & & & & & \\ & & 1 & 1 & & & & \\ & & 1 & -1 & & & & \\ & & & & 1 & 1 & & \\ & & & & 1 & -1 & & \\ & & & & & & 1 & 1 \\ & & & & & & 1 & -1 \end{bmatrix}}_{B_1}.\underbrace{\begin{bmatrix} 1 & & & & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & & & & 1 \end{bmatrix}}_{\left(\left[P_{(8)}\right]\right)^{-1}}.\underbrace{\begin{bmatrix} 1 & & & & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & & & & 1 \end{bmatrix}}_{\left[P_{(8)}\right]}.$$

$$\frac{1}{\sqrt{2}}\underbrace{\begin{bmatrix} 1 & 1 & & & & & & \\ 1 & -1 & & & & & & \\ & & 1 & 1 & & & & \\ & & 1 & -1 & & & & \\ & & & & 1 & 1 & & \\ & & & & 1 & -1 & & \\ & & & & & & 1 & 1 \\ & & & & & & 1 & -1 \end{bmatrix}}_{B_0}.\underbrace{\begin{bmatrix} 1 & & & & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & & & & 1 \end{bmatrix}}_{\left[BR_{(8)}\right]} \tag{47}$$

Llavors, les noves etapes són:

$$B_2^{'} = B_2.\left(\left[P_{(8)}\right]^2\right)^{-1} \quad ; \quad B_1^{'} = \left[P_{(8)}\right]^2.B_1.\left(\left[P_{(8)}\right]\right)^{-1} \quad ; \quad B_0^{'} = \left[P_{(8)}\right]B_0 \tag{48}$$

Obtenint la següent expressió amb les noves etapes:

$$F_{(8)} = \frac{1}{\sqrt{2}} \underbrace{\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 & e^{-j\frac{2\pi}{8}} \\ 1 & -e^{-j\frac{2\pi}{8}} \end{bmatrix} \cdot \begin{bmatrix} 1 & -j \\ 1 & j \end{bmatrix} \cdot \begin{bmatrix} 1 & e^{-j\frac{6\pi}{8}} \\ 1 & -e^{-j\frac{6\pi}{8}} \end{bmatrix}}_{B_2} \frac{1}{\sqrt{2}} \underbrace{\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 & -j \\ 1 & j \end{bmatrix} \cdot \begin{bmatrix} 1 & -j \\ 1 & j \end{bmatrix}}_{B_1} \frac{1}{\sqrt{2}} \underbrace{\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}}_{B_0} \cdot$$

$$\underbrace{\begin{bmatrix} 1 & & & & & & & \\ & & & 1 & & & & \\ & & 1 & & & & & \\ & & & & & 1 & & \\ & 1 & & & & & & \\ & & & & 1 & & & \\ & & & 1 & & & & \\ & & & & & & & 1 \end{bmatrix}}_{BR_{(8)}} \tag{49}$$

Delmació en freqüència

S'ha vist que a partir de la propietat de delmació en temps i introduint matrius de permutació i les seves inverses, la transformada discreta de Fourier és descomposada en etapes idèntiques formades per unes estructures bàsiques anomenades papallones. A continuació es vol aconseguir el mateix, però per la transformada inversa discreta de Fourier a partir de la propietat de delmació en freqüència.

Partint de la següent propietat de delmació en freqüència:

$$F_{(N)} = \frac{1}{\sqrt{2}} [P_{(N)}]^T \begin{bmatrix} F_{(N/2)} & \\ & F_{(N/2)} \end{bmatrix} \cdot \begin{bmatrix} I_{(N/2)} & D_{(N/2)} \\ I_{(N/2)} & -D_{(N/2)} \end{bmatrix} \tag{50}$$

On *T* és la transposta. El ser $F_{(N)}$ una matriu hermítica es compleix que la seva inversa és igual a la seva hermítica [44]. Aplicant aquesta propietat a l'expressió anterior queda com:

$$F_{(N)}^{-1} = (F_{(N)})^H = \frac{1}{\sqrt{2}} \left([P_{(N)}]^T \begin{bmatrix} F_{(N/2)} & \\ & F_{(N/2)} \end{bmatrix} \cdot \begin{bmatrix} I_{(N/2)} & D_{(N/2)} \\ I_{(N/2)} & -D_{(N/2)} \end{bmatrix} \right)^H = \frac{1}{\sqrt{2}} \left(\begin{bmatrix} I_{(N/2)} & D_{(N/2)} \\ I_{(N/2)} & -D_{(N/2)} \end{bmatrix}^H \begin{bmatrix} F_{(N/2)} & \\ & F_{(N/2)} \end{bmatrix}^H ([P_{(N)}]^T)^H \right) =$$

$$\frac{1}{\sqrt{2}} \begin{bmatrix} I_{(N/2)} & D_{(N/2)}{}^H \\ I_{(N/2)} & -D_{(N/2)}{}^H \end{bmatrix} \cdot \begin{bmatrix} F_{(N/2)}{}^H & \\ & F_{(N/2)}{}^H \end{bmatrix} [P_{(N)}] \tag{51}$$

On *H* és la hermítica.

Segons aquestes propietats s'arriba a una estructura idèntica per la FFT i la FFT amb la diferència que els pesos que s'obtenen utilitzant la delmació en freqüència són els mateixos que en la delmació en temps però conjugats. Llavors introduint les mateixes matrius de permutació utilitzades en delmació en temps, s'aconsegueix el mateix diagrama de flux.

Referències

[1] P.Martí, *Algorismes de detecció i recuperació de referència per a modulacions transformades*, (in Catalan) PhD Thesis, UPC, 2001.

[2] R.Van Nee, R. Prasad, *OFDM for Wireless Multimedia Communications*, Artec House Publishers, 2000.

[3] J. Heiskala, J. Terry, *OFDM Wireless LANs: a theoretical and practical guide*, Sams Publishing, 2002.

[4] TS 101 475, *Broadband Radio Acces Networks (BRAN), HiperLAN/2; Phisical Layer (PHY)*, http://www.etsi.org, 2001.

[5] TS 101 761-1, *Broadband Radio Access Networks (BRAN);HIPERLAN Type 2;Data Link Control (DLC) Layer; Part 1: Basic Data Transport Functions*, http://www.etsi.org, 2001.

[6] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHZ Band*, IEEE Std 802.11a-1999.

[7] Ph. Rouzet, L. Bisdounis, *Exploration of the HIPERLAN/2 Standard and Comparison with Other wireless LAN Standards*", IST-2000-30093, Jan. 2002.

[8] S. B. Weinstein, P. M. Ebert, *Data Transmission by Frequency Division Multiplexing Using the Discrete Fourier Transform*, IEEE Trans. Comm., COM-19:628–634, Oct. 1971.

[9] K. Masselos, S. Blionas, T. Rautio, *Reconfigurability requeriments of wireless communication systems*, IEEE Workshop on Heterogeneous Reconfigurable Systems on Chip, Chances, Application, Trends, April 2002.

[10] C.J. Booth and G.P. Kurpis, *The New IEEE Standard Dictionary of Electrical and Electronics Terms*, fifth ed. New York: IEEE, 1993.

[11] F. Kordon and Luqi, *An Introduction to Rapid System Prototyping, IEEE Transactions on Software Enginnering*", vol. 28, nº.9, September 2002.

[12] IP cores, www.actel.com/products/ip/ ; www.keil.com/dd/ipcores.asp ; www.alatek.com/pages/products/ip_cores.asp.

[13] Sistemes de prototipatge Hardware, http://www.ikos.com/; www.cadence.com

[14] Plataformes de disseny SoC, www.ceva-dsp.com/

[15] Plataformes educacionals, www.xess.com/

[16] Plataformes de computació reconfigurable, www.xilinx.com/ ; www.actel.com/

[17] System Generator, ISE i Chiscope, http://www.xilinx.com

[18] DSP Builder, http://www.altera.com

[19] Matlab/Simulink, http://www.mathworks.com

[20] **M. Serra**, *Adaptació d'una metodologia de codisseny HW/SW pel prototipat de sistemes encastats reactius*, tesina, UAB, 2001.

[21] **M. Serra**, J. Tirado, J. Saiz, J. Carrabina, *Experiments in rapid prototyping of reactive systems with Polis*, XVI Conference on design of circuits and integrated systems, Porto, 2001.

[22] J. Tirado, **M. Serra**, A. Portero, Q. Saiz, L. Ribas, J. Carrabina, *Rapid Prototyping Platform for Reactive Systems with a POLIS-based HW-SW Co-design Approach*, IP Based Design 2000, Grenoble, 2000.

[23] Q. Saiz, **M. Serra**, J. Tirado, A. Portero, L. Ribas, J. Carrabina, *Un camino desde el codiseño HW/SW hasta las plataformas de prototipado para computación configurable*, JCRA2001, Alicante, 2001.

[24] OCAPI, http://www.imec.be/design/ocapi/
[25] HandelC, http://www.celoxica.com/
[26] SystemC, http://www.systemc.org
[27] SystemVerilog, http://www.systemverilog.org/
[28] Ptolemy II, http://ptolemy.eecs.berkeley.edu/ptolemyII/
[29] Signal Processing Worksystem (SPW), http://www.coware.com
[30] COSSAP, http://www.synopsis.com
[31] DSP Station, http://www.mentor.com/
[32] Advanced Design System (ADS), http://eesof.tm.agilent.com/
[33] M. Gokhale, J. Stone, and J. Arnold, *Stream-Oriented FPGA Compunting in the streams-C High Level Language*, in Proc. FCCM00, B. Hutchings (ed), IEEE Computer Society Press, pp. 49-56, 2000.
[34] Nallatech, http://www.nallatech.com.
[35] Alpha-Data, http://www.alpha-data.com
[36] Annapolis MicroSystems, http://www.annapmicro.com
[37] Lyrtech, http://www.lyrtech.com
[38] **M. Serra**, O. Navas, J. Escrig , M. Bonamusa , P. Martí, J. Carrabina, *Metodología de prototipado rápido desde Matlab: herramientas visuales para flujo de datos*, IV Jornadas sobre computación reconfigurable y aplicaciones, Bellaterra(Barcelona), 2004.
[39] J. Sala, *Una Metodologia d'Avaluació de Sistemes de Processament de Senyal a Nivell Binari*, I Jornades de Codisseny Hardware-Software, Vic, 2003.
[40] K. Keutzer, A.R. Newton, J.M Rabaey, and A. Sangiovanni-Vincentelli, *System-Level Design: orthogonalization of concerns and platform-based design*, IEEE Trans. On Computer-Aided Design of Integrated Circuits and Systems, vol. 19, pp.1523-43, 2000.
[41] C.Thorpe*, OFDM Wireless Systems Simulation Using Simulink*, International DSP Conference, Stuttgart, May 1998.
[42] J.Medbo, P.Schramm, *Channel Models for HIPERLAN/2 in Different Indoor Scenarios*, ETSI EP BRAN 3ERIO85B, March 1998.
[43] M. Engels, *Wireless OFDM Systems,* Kluwer Academic Publishers, United States of America, 2002.
[44] J. R. Schott, *Matrix Analysis for Statistics*. Wiley, 1997.
[45] J.W. Cooley and J.W. Tukey, *An algorithm for the machine computation of the complex Fourier series*, Mathematics ofComputation, vol. 19, pp. 297–301, Apr. 1965.
[46] M.A. Sancho, F.E. Angarita, T. Sansaloni, A. Perez-Pascual, *FFT de 64 Puntos para Redes de Datos Inalámbricas Basadas en OFDM*, IV Jornadas de Computación Reconfigurable y Aplicaciones, UAB, 2004.
[47] **M. Serra**, P. Martí, J. Carrabina, *IFFT/FFT Core Architecture with an Identical Stage Structure for wireless LAN Communications*. V IEEE Worshop on Signal Processing Advances in Wireless Commnunications, Lisbon (Portugal), 2004.
[48] P. Martí, D. Novo, **M. Serra**, J. Carrabina, *Diseño de un bloque FFT/IFFT con alto grado de paralelismo mediante FPGA*, I Jornades de Codisseny Hardware Sofware, Vic, 26 y 27 de junio 2003.
[49] P. Martí, D. Novo, **M. Serra**, J. Carrabina, *A matrix representation method to develop FFT/IFFT parallel architectures*, 7th International OFDM- Workshop, Hamburg, 10,11-09- 2002.
[50] P. Martí, D. Novo, **M. Serra**, J. Carrabina, *Diseño de un bloque FFT/IFFT con alto grado de paralelismo mediante FPGA*, II Jornadas sobre computación reconfigurable y aplicaciones, Almuñecar, 18,20-09- 2002.
[51] Xilinx, *Fast Fourier Transform v2.0, DS260 (v2.0)*, July 14, 2003.

[52] P. H. Moose, *A technique for orthogonal frequency division multiplexing frequency offset correction*. IEEE Trans. on Comm., Vol. 42, No. 10:2908–2914, Oct. 1994.
[53] T.M. Schmidl, D.C. Cox, *Robust Frequency Offset Estimator for OFDM. IEEE Transactions on Communications*. Vol 45, no. 12, 1997.
[54] T. Keller, L. Piazzo, P. Mandarini, L. Hanzo, *OFDM Synchronization Techniques for Frequency-Selective Fading Channels*. IEEE Journal on Selected Areas in Communications. Vol 42, no. 10, octubre 1994.
[55] S. Johansson, P. Nilsson, M. Torkelson, *Implementation of an ODFM Synchronization Algorithm*, Circuits and Systems, 42nd Midwest Symposium on ,Vol. 1 , pp. 228-231, 1999.
[56] V.Almenar, S. Abedi, R. Tafazolli, *Synchronization Techniques for HiperLAN/2*, IEEE VTS 54th Vehicular Technology Conference, Vol. 2, pp. 762-766, 2001.
[57] A. Berno, N. Laurenti, *Time and Frequency Synchronization for Hiperlan/2*, Dipartimento di Elettronica ed Informatica, Università di Padova, 2001.
[58] J.-J. Van de Beek, M. Sandell, P.O. Börjesson, *ML Estimation of Time and Frequency offsets in OFDM Systems*, IEEE trans. signal processing, Vol. 45, pp. 1800-1805, 1997.
[59] E. Mumbrú, **M. Serra**, D. Novo, P. Martí, J. Carrabina, *Sistema de Sincronització per l'estándar Hiperlan/2*, I Jornades de Codisseny Hardware Sofware, Vic, 26 y 27 de junio 2003.
[60] R. Andraka, *A survey of CORDIC algorithms for FPGA based computers*, Sixth International Symposium on Field Programmable Gate Arrays, Monterey,CA, 1998.
[61] P.Hung, H. Fahmy, O. Mencer and M.J.Flynn, *Fast Division Algorithm with Small Lookup table*, Conference Record of the Thirty-Third Asilomar Conference on Signals, Systems and Computers, Vol.2, May pp 1465-1468,1999.
[62] S.F. Oberman and M.J.Flynn, *Division Algorithms and Implementations*, IEEE Transactions on Computers, Vol. 46, No.8, pp 833-854, Aug. 1997.
[63] **M. Serra**, P. Martí, J. Carrabina, *Implementation of a Channel Equalizer for OFDM Wireless LANs*. 15th IEEE International Workshop on Rapid System Prototyping, Geneva, Switzerland, 2004.
[64] **M.serra**, D. Novo, P. Martí, J. Carrabina, *Equalitzador de Canal per a Modulacions OFDM Segons l'Estàndard HiperLAN/2*, I Jornades de Codisseny Hardware-Software, Vic, 2003.
[65] A.J. Viterbi, *CDMA*, *Principles of Spread Spectrum Communication*, Addison-Wesley Publis-hing Company, 1995.
[66] J. G. Proakis, *Digital Communications*, Third Edition, McGraw-Hill International Editions, 1995.
[67] J. Ordeix, P. Martí, **M. Serra**, J. Carrabina. *Arquitectura de un decodificador convolucional a partir del algoritmo de Viterbi*. IV Jornadas sobre computación reconfigurable y aplicaciones, Bellaterra(Barcelona), 2004
[68] J. Ordeix, *Disseny i prototipat d'un IP per a mòduls de comunicació SoC: Descodificador convolucional a partir de l'algorisme de Viterbi*, Tesina, 2002, UVic-UAB.
[69] J. Ordeix, S. Parés. P. Martí, J. Carrabina, *Descodificador convolucional a partir de l'algorisme de Viterbi*, I Jornades de Codisseny HW-SW, Vic, 2003.
[70] Li Shang, A. Kaviani i K. Bathala, *Dynamic Power Consumption in Virtex-II FPGA family*, FPGA'02, Monterey, California, 2002.

[71] **M. Serra**, P. Martí, J. Bover, J. Carrabina, *Prototipado rápido de un sistema WLAN*. IV Jornadas sobre computación reconfigurable y aplicaciones, Bellaterra (Barcelona), 2004.

[72] **M. Serra**, J. Ordeix, P. Martí, J. Carrabina, *OFDM Demonstrator: Transmitter*, 7th International OFDM- Workshop, Hamburg, 10,11-09- 2002.

[73] **M. Serra**, X. Rafael, J.Ordeix, P. Martí, J. Carrabina, *Prototipo demostrador de OFDM: Transmisor*, II Jornadas sobre computación reconfigurable y aplicaciones, Almuñecar, 18,20-09- 2002.

[74] J. Hwang, *FPGA Code Portability for Signal Processing in Software Defined Radios*, Xilinx, San José (CA), 2004.

[75] Conversió de FPGA a ASIC: XpressArray, http://www.amis.com/; ASICentrum, http://www.asicentrum.cz/

[76] W. Eberle, V. Derudder, L. Van der Perre, G. Vanwijnsberhe, M. Vergara, L. Deneire, B. Gyselinckx, M. Engels, I. Bolsens, *80-Mb/s QPSK and 72-Mb/s 64-QAM Flexible and Scalable Digital OFDM Transceiver ASICs for Wireless Local Area Networks in the 5-GHz Band*. IEEE International Solid State Conference, Vol. 36, pp. 1829-1838, 2001.

[77] P. Ryan, T. Arivoli, L. de Souza, G. Foyster, R. Keaney, T. McDermott, A. Moini, S. Al-Sarawi, U. Parker, G. Smith, N. Weste, G. Zyner, *A Single Chip PHY COFDM Modem for IEEE802.11a with Integrated ADCs and DACs*, Solid-State Circuits Conference. Digest of Technical Papers. ISSCC. 2001 IEEE International, Pages:338 - 339, 463, 5-7 Feb. 2001.

[78] J. Kneip, M. Weiss, W. Drescher, V. Aue, J. Strobel, T. Oberthür, M. Bolle, G. Fettweis, *Single Chip Programable Base Band ASSP for 5GHz Wireless LAN Applications*, IECIE Trans. Electron. Vol. E-85 nº2, February 2002.

[79] RapidChip Platform ASICs vs FPGAs, www.lsilogic.com/