UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Implementación VLSI del algoritmo de proyecciones sucesivas para detección de sistemas MIMO

Tesis Doctoral

Autor:
José Marín-Roig Ramón

Directores:
Dr. Javier Valls Coquillat
Dr. Vicenç Almenar Terré

Valencia, 10 de Diciembre de 2015

Resumen

La insaciable demanda de ancho de banda de comunicación por parte de los usuarios finales, unido al abaratamiento de los terminales y de los servicios de telecomunicación han provocado un crecimiento espectacular del mercado de las comunicaciones inalámbricas en estos últimos años.

Las entidades responsables, a nivel internacional, de la estandarización tecnológica han sabido acompañar y guiar este crecimiento redactando normas como LTE (Long Term Evolution), IEEE 802.11 (WiFi) e IEEE 802.16 (WiMax) o las redes 3G o 4GPP. Todas ellas comparten como denominador común, para la mejora de la eficiencia espectral, el uso de las tecnologías MIMO, que utiliza múltiples antenas en emisor y receptor, y el uso de esquemas de modulación elevados como 256-QAM, introducido en la revisión 12 del estándar 3GPP-LTE.

Bajo esta perspectiva de grandes ganancias en la eficiencia espectral, no es de extrañar que la tecnología MIMO haya sido incorporada en los estándares mencionados anteriormente. No obstante, conseguir estas ganancias no es trivial, hasta el punto de que la implementación VLSI de esta tecnología se ha convertido en un reto.

En esta tesis se ha realizado un estudio exhaustivo de diferentes detectores MIMO, fijando el punto de mira en aquellos pertenecientes a las dos familias que muestran mejores características para su implementación VLSI: cancelación sucesiva de interferencias (detector VBLAST) y basados en búsqueda en árbol (detector KBest). Aunque inicialmente las prestaciones alcanzadas por los segundos (KBest) son muy superiores a las de los primeros (VBLAST), la reciente aparición en la literatura especializada del algoritmo de proyecciones sucesivas (SPA) abre la puerta al desarrollo de un nuevo detector, perteneciente a la familia de los detectores de cancelaciones sucesivas de interferencias (SIC), que pueda competir en prestaciones con los detectores KBest.

La tarea de desarrollar un detector, cuya implementación *hardware* sea viable y competitiva, tomando como punto de partida el algoritmo original publicado, no es trivial. Los detectores basados en el algoritmo SPA ofrecen como valor añadido, frente a otros detectores SIC, la posibilidad de generar de forma iterativa distintas soluciones. Sin embargo sus autores no establecen ningún método para controlar las repeticiones en las iteraciones, ni se estudia el número de iteraciones requeridas para aproximar la solución ML con un coste computacional razonable.

En esta tesis se aportan las claves algorítmicas necesarias que hacen viable y competitiva la implementación *hardware* del algoritmo SPA. En particular, se han desarrollado dos mecanismos de control de repeticiones: Simplified-ESPA (SESPA) y Table-ESPA (TESPA), y se han adaptado los mecanismos de obtención de salidas *hard output* y *soft output*, existentes en la literatura, a este algoritmo.

Se ha diseñado la primera arquitectura VLSI para el algoritmo SPA, siendo ésta altamente flexible, en el sentido de que se adapta a diferentes condiciones de transmisión y cumple con las últimas especificaciones publicadas en los estándares WiMAX y LTE. La flexibilidad de la arquitectura permite seleccionar diferentes configuraciones de antenas en transmisión y recepción, desde 2x2 hasta 4x4, diferentes esquemas de modulación desde QPSK hasta 256QAM, controla el balance entre tasa de transmisión y las prestaciones BER/FER y ofrece las decisiones *soft output* y *hard output*.

Finalmente, con esta arquitectura se ha realizado la implementación de los detectores SESPA y TESPA, con salidas *soft output* y *hard output*, en los dispositivos FPGA y ASIC. Estos detectores han sido evaluados y comparados con los mejores publicados en la literatura especializada, consiguiendo la tasa de pico máxima de 465 Mbps para el detector SESPA 4x4 256-QAM, en un área de $3.83\,mm^2$ con una tecnología de $90\,nm$. Los detectores implementados ofrecen como valor añadido, además de la alta configurabilidad, la posibilidad de decodificar 256QAM sin incrementar el área. Esta característica es altamente competitiva con los detectores no lineales basados en KBest, que son muy sensibles, en cuanto a tasa de decodificación y área se refiere, con el esquema de modulación seleccionado. Además, los detectores basados en ESPA alcanzan unas prestaciones FER (*soft output*) claramente competitivas con los detectores KBest, debido a la mayor calidad del LLR generado por el ESPA. La comparación con otras arquitecturas flexibles seleccionadas demuestra que los detectores SESPA y TESPA ofrecen la mayor configurabilidad de parámetros de transmisión y el mejor equilibrio entre área, prestaciones BER y tasa detección.

Abstract

The insatiable demand for bandwidth of communication on the part of end-users, linked to the lowering the price of the terminals and in telecommunication services have led to a spectacular growth of the wireless communications market in recent years.

Those entities that are responsible, at the international level, of the technological standardization have known to guide this growth writing standards as LTE (Long Term Evolution), IEEE 802.11 (WiFi) and IEEE 802.16 (WiMax) or 3G networks or 4GPP. They all share a common denominator, for the improvement of the spectral efficiency, the use of MIMO technologies, which uses multiple antennas on transmitter and receiver, and the use of high modulation schemes as 256-QAM, introduced in revision 12 of the standard 3GPP-LTE.

Under this perspective of great gains in the spectral efficiency, it is not surprising that MIMO technology has been incorporated into the standards mentioned above. However, achieving these gains is not trivial, to the extent that the VLSI implementation of this technology has become a challenge.

In this thesis has undertaken a comprehensive study of different MIMO detectors, studying those belonging to the two families that show best features for being implemented in VLSI technology: successive interference cancellation (VBLAST detector) and based on a search in tree (KBest detector). Although initially the benefits achieved by the seconds (KBest) are far superior to those of the first (VBLAST), the recent appearance in the specialized literature of the Successive Projections Algorithm (SPA) opens the door to the development of a new detector, belonging to the family of the detectors of Successive Interference Cancellations (SIC), which will be able to compete in performance with the KBest detectors.

The task of developing a detector, whose implementation hardware will be viable and competitive, taking as a starting point the original published

algorithm, is not trivial. Detectors based on the algorithm SPA offer as added value, compared to other detectors SIC, the possibility of generating iteratively different solutions. However their authors do not set any method to control the repeats in the iterations, or studying the number of iterations required to approximate the ML solution with a reasonable computational cost.

In this thesis, we provide the necessary algorithmic keys that make viable and competitive the hardware implementation of the SPA algorithm. In particular, two mechanisms of control of repetitions have been developed: Simplified-ESPA (SESPA) and Table-ESPA (TESPA), and the mechanisms for obtaining hard and soft output, existing in the literature, have been adapted to this algorithm. It has designed the first VLSI architecture for the SPA algorithm, being highly flexible, in the sense that adapts to different conditions of transmission and complies with the latest published specifications in the WiMAX and LTE standards. The flexibility of the architecture allows you to select different configurations of antennas in transmission and reception, from 2x2 to 4x4, different modulation schemes from QPSK until 256-QAM, controls the balance between transmission rate and the benefits BER/FER and offers the soft output and hard output decisions.

Finally, with this architecture has been implemented the SESPA and TESPA detectors, with soft output and hard output, in FPGA and ASIC technology. These detectors have been evaluated and compared to the best published in the specialized literature, achieving a peak rate of 465 Mbps for the detector SESPA 4x4 256-QAM, with an area of $3.83\,mm^2$ with a $90\,nm$ technology. The detectors implemented offer as added value, in addition to the high configurability, the ability to decode 256-QAM without increasing the area. This feature is highly competitive with the non-linear detectors based on KBest, which are very sensitive, in regard to decoding rate and area, with the selected modulation scheme. In addition, the detectors based on ESPA reach a FER performance (soft output) clearly competitive with KBest detectors, due to a higher quality of the LLR generated by the ESPA. The comparison with other flexible architectures selected shows that the SESPA and TESPA detectors offer the greater configurability of transmission parameters and the best balance between area, BER performance and detection rate.

Resum

La insaciable demanda d'ample de banda de comunicació per part dels usuaris finals, unit a l'abaratiment dels terminals i dels servicis de telecomunicació han provocat un creixement espectacular del mercat de les comunicacions sense fils en aquests últims anys.

Les entitats responsables, a nivell internacional, de l'estandardització tecnològica han sabut acompanyar i guiar aquest creixement redactant normes com LTE (Long Term Evolution), IEEE 802.11 (WiFi) i IEEE 802.16 (WiMax) o les xarxes 3G o 4GPP. Totes elles comparteixen com denominador comú, per a la millora de l'eficiència espectral, l'ús de les tecnologies MIMO, que utilitza múltiples antenes en emissor i receptor, i l'ús d'esquemes de modulació elevats com 256-QAM, introduït en la revisió 12 de l'estàndard 3GPP-LTE.

Baix esta perspectiva de grans guanys en l'eficiència espectral, no és d'estranyar que la tecnologia MIMO hi haja estat incorporada en els normatives mencionats anteriorment. No obstant això, aconseguir aquests guanys no és trivial, fins l'extrem que la implementació VLSI d'aquesta tecnologia s'ha convertit en un repte.

En aquesta tesi s'ha realitzat un estudi exhaustiu de diferents detectors MIMO, fixant el punt de mira en aquells que pertanyen a les dos famílies que mostren millors característiques per a la seua implementació VLSI: cancel·lació successiva d'interferències (detector VBLAST) i els basats en recerca en arbre (detector KBest). Encara que inicialment les prestacions aconseguides pels segons (KBest) són molt superiors a les dels primers (VBLAST), la recent aparició en la literatura especialitzada de l'algoritme de projeccions successives (SPA) permet el desenvolupament d'un nou detector, que pertany a la família dels detectors de cancel·lacions successives d'interferències (SIC), que puga competir en prestacions amb els detectors KBest.

La tasca de desenvolupar un detector, amb implementació hardware via-

ble i competitiva, a partir de l'algoritme original publicat, no és trivial. Els detectors basats en l'algoritme SPA ofereixen com a valor afegit, front a altres detectors SIC, la possibilitat de generar de forma iterativa distintes solucions. No obstant, els seus autors no estableixen cap mètode per controlar les repeticions en les iteracions, ni s'estudia el nombre d'iteracions requerides per a aproximar la solució ML amb un cost computacional raonable.

En aquesta tesi s'aporten les claus algorítmiques necessàries que fan viable i competitiva la implementació hardware de l'algoritme SPA. En particular, s'han desenvolupat dos mecanismes de control de repeticions: Simplified-ESPA (SESPA) i Table-ESPA (TESPA), i s'han adaptat els mecanismes d'obtenció d'eixides *hard-output* i *soft-output*, existents en la literatura, a aquest algoritme.

S'ha dissenyat la primera arquitectura VLSI per a l'algoritme SPA, sent aquesta altament flexible, en el sentit de que s'adapta a diferents condicions de transmissió i acompleix les últimes especificacions publicades en els estàndards WiMax i LTE. La flexibilitat de l'arquitectura permet seleccionar diferents configuracions d'antenes en transmissió i recepció, des de 2x2 fins 4x4, diferents esquemes de modulació des de QPSK fins 256-QAM, controla el balanç entre taxa de transmissió i les prestacions BER/FER i ofereix les decisions *hard output* i *soft output*.

Finalment, amb l'arquitectura proposta s'ha realitzat la implementació dels detectors SESPA i TESPA, amb eixides *hard output* i *soft output*, en els dispositius FPGA i en ASIC. Aquests detectors han segut valorats i comparats amb els millors publicats en la literatura especialitzada, i s'ha aconseguint la taxa de pic màxim de 465 Mbps per al detector SESPA 4x4 256-QAM, dins una àrea de 3.83 mm^2 en una tecnologia de 90 nm. Els detectors implementats ofereixen com a valor afegit, a més de l'alta configurabilitat, la possibilitat de decodificar 256-QAM sense incrementar l'àrea. Esta característica és altament competitiva en els detectors no lineals basats en KBest, que són molt sensibles, en relació a taxa de decodificació i a l'àrea del circuit, a l'esquema de modulació seleccionada. A més a més, els detectors basats en ESPA aconsegueixen unes prestacions FER (*soft output*) clarament competitives amb els detectors KBEST, degut a la major qualitat del LLR generat per l'ESPA. La comparació amb altres arquitectures flexibles seleccionades demostra que els detectors SESPA i TESPA ofereixen una major configurabilitat de paràmetres de transmissió i un millor equilibri entre l'àrea del circuit, les prestacions BER i la taxa de detecció.

Agradecimientos

No me cabe la menor duda de que este trabajo, desde su génesis hasta su finalización, jamás lo habría podido desarrollar sin la dirección que he tenido. Me consta que esto es una suerte de la que no todos los doctorandos disfrutan. El trabajo y dedicación de mis dos directores, cada uno en su área, excede con creces el mérito que para ellos supone esta tesis. Javi y Vicenç, gracias de corazón.

Todo gran trabajo requiere de buenos colaboradores. Compañeros que me han facilitado el trabajo ayudándome en tareas de investigación como José Correcher y Fabian Angarita, en tareas docentes como María José Canet o en tareas de gestión como Asun Pérez. Me encantaría poderos devolver lo que habéis hecho por mí de la misma manera, colaborando en vuestros trabajos, pero mientras llega ese día, os adelanto mis gracias de corazón.

Podría escribir una página entera con nombres de compañeros, amigos y familiares que durante estos últimos años siempre han recibido la misma respuesta, cuando me preguntaban sobre el estado de mi tesis: "ya está casi terminada". A todos vosotros que me habéis alentado con muchas pequeñas dosis de ánimo, gracias de corazón.

Con la perspectiva de los años es fácil olvidar a los que han invertido tantos esfuerzos e ilusiones en que yo sea lo que soy. Ojalá mis hijos puedan mostrarme su agradecimiento en el futuro, como yo lo hago cada día con ellos. Gracias papá, gracias mamá. Gracias de corazón.

Finalmente, la mención especial se la lleva la mejor madre, la que más veces ha recibido el "ya está casi terminada" por respuesta, la mejor colaboradora, la que más me facilitado las tareas de casa y la educación de mis hijos y la mejor directora, la que mejor ha dirigido mi vida personal. La mujer de mi vida, Mayte. Gracias. Ella sabe que son de corazón.

Índice General

Lista de Figuras

Lista de Tablas

Lista de Algoritmos

Lista de Acrónimos

3GPP	*3rd Generation Partnership Project*
4G	*4rd Generation*
APP	*A Posteriori Probability*
ASIC	*Application-Specific Integrated Circuit*
AWGN	*Additive White Gaussian Noise*
BER	*Bit Error Rate*
BPSK	*Binary Phase Shift Keying*
BSPA	*Basic Successive Projection Algorithm*
CDMA	*Code Division Multiple Access*
CMOS	*Complementary Metal-Oxide-Semiconductor*
CSMA/CA	*Carrier Sense Multiple Access with Collision Avoidance*
ESPA	*Extended Successive Projection Algorithm*
FEC	*Forward Error Correction*
FER	*Frame Error Rate*
FPGA	*Field Programmable Gate Array*
IEEE	*Institute of Electrical and Electronics Engineers*
LDPC	*Low Density Parity Check*

LLR	*Log-Likelihood Ratio*
LR	*Lattice Reduction*
LSB	*Least Significant Bit*
LUT	*Look-Up Table*
LTE	*Long Term Evolution*
MCU	*Metric Computation Unit*
MIMO	*Multi-Input Multi-Output*
ML	*Maximum Likelihood*
MMSE	*Minimum Mean Square Error*
MSB	*Most Significant Bit*
OSIC	*Ordered Successive Interference Cancellation*
OFDM	*Orthogonal Frequency Division Multiplexing*
PSU	*Path Select Unit*
QAM	*Quadrature Amplitude Modulation*
QoS	*Quality of Service*
QPSK	*Quadrature Phase Shift Keying*
RCU	*Repetition Control Unit*
ROM	*Read-Only Memory*
RVD	*Real Value Descomposition*
SD	*Sphere Decoding*
SIC	*Successive Interference Cancellation*
SESPA	*Simplified Extended Successive Projection Algorithm*
SNR	*Signal-to-Noise Ratio*
TESPA	*Table Extended Successive Projection Algorithm*
TBU	*Traverse Brach Unit*
VHDL	*Very High Speed Integrated Circuit Hardware Description Language*

SPA	*Successive Projection Algorithm*
VLSI	*Very Large Scale Integration*
V-BLAST	*Vertical-Bell Laboratories Layered Space-Time*
WIFI	*Wireless Fidelity*
WLAN	*Wireless Local Area Network*
WIMAX	*Worldwide Interoperability for Microwave Access*
ZF	*Zero Forzing*

Capítulo 1

Introducción

El mercado de las comunicaciones inalámbricas es, sin duda, uno de los que más rápidamente ha crecido en el mundo en los últimos 20 años. Este crecimiento se basa fundamentalmente en tres razones:

- La insaciable demanda de ancho de banda de comunicación por parte de los usuarios finales, que no sólo solicitan más servicios sino que los exigen de mayor calidad. Según los datos publicados por la Fundación Telefónica en su informe del 2014, existen en el mundo casi tantos teléfonos móviles (6.800 millones) como personas (7.100 millones).

- El abaratamiento de los terminales de usuario, como los teléfonos inteligentes, y la aparición de nuevos dispositivos como las tablets, es consecuencia de la enorme demanda por parte de los usuarios y del inevitable crecimiento de la competencia entre los fabricantes.

- El abaratamiento de los servicios de telecomunicación, especialmente del acceso a internet a través de la banda ancha móvil debido a la aparición de nuevas operadoras de telecomunicación. El desembarco en los mercados de la telefonía móvil provocó la aparición de nuevas compañías rompiendo el régimen de cuasi monopolio de operadoras tradicionales.

Si bien estas tres razones atienden únicamente a parámetros estrictos de mercado, oferta y demanda, no puede pasar desapercibido un cuarto motivo que ha hecho que este crecimiento sea posible: los esfuerzos de estandarización tecnológica. Los estándares de comunicación inalámbrica a

través del proyecto "Long-Term Evolution" (LTE) han dotado de la eficiencia necesaria a las comunicaciones. Desde el 802.11 (WIFI) hasta el 802.16 (WIMAX) ó las redes 4G ó 3GPP ("3rd Generation Partnership Project"), todas comparten como denominador común el uso de las tecnologías MIMO (Multiple-Input-Multiple-Output) y de esquemas de modulación elevados (256-QAM en la revisión 12 del estándar 3GPP-LTE), como herramienta necesaria para conseguir alta eficiencia espectral, altas tasas de transmisión de datos y robustez en la comunicación inalámbrica.

1.1. Tecnología MIMO

Los sistemas de comunicación MIMO emplean múltiples antenas en ambos lados, transmisor y receptor para incrementar las prestaciones de comunicación. Desde el punto de vista teórico de la información, incrementar el número de antenas básicamente permite alcanzar mayor eficiencia espectral comparada con un sistema SISO ("Single-Input-Single-Output").Esta mayor capacidad de transmisión se fundamenta en tres ganancias [1]:

- *Ganancia de Array*, para captar un mayor porcentaje de la potencia transmitida en el receptor, lo que permite extender el alcance del sistema de comunicación y suprimir interferencias.
- *Ganacia de Diversidad*, que tiene en cuenta el efecto de las variaciones de canal, denominadas "fading", incrementando la fiabilidad del enlace y la calidad del servicio (QoS).
- *Ganancia de Multiplexación*, que permite un incremento lineal de la eficiencia espectral y de la tasa de transmisión de pico, como consecuencia de la transmisión concurrente de flujos múltiples de datos en la misma banda de frecuencias. El número de flujos de datos en paralelo está limitado por el número de antenas transmisoras y/o receptoras.

Bajo esta perspectiva de enormes ganancias, no es de extrañar que la tecnología MIMO haya sido incorporada en los estándares mencionados anteriormente. No obstante, conseguir estas ganancias no es trivial, hasta el punto de que la implementación de esta tecnología se ha convertido en un reto.

Una consecuencia directa de la multiplexación espacial es la mayor complejidad del procesado de señal, fundamentalmente en el receptor y a veces en el transmisor. El incremento lineal de la eficiencia espectral con el número de antenas en emisor y receptor se traduce en un incremento más que lineal de la complejidad del decodificador, incluso utilizando los algoritmos más básicos. Para explotar completamente el potencial de la tecnología multiantena se requieren algoritmos que incluso tienen una complejidad superior

a lo que se puede implementar en los circuitos integrados de hoy en día, o dicho de otro modo a lo que es económicamente viable.

Por lo tanto el reto de la tecnología MIMO es el diseño de algoritmos de baja complejidad en los receptores y el desarrollo de sus correspondientes arquitecturas VLSI.

1.2. Estado del arte

Desde un punto de vista cronológico la evolución de los algoritmos de decodificación de los sistemas MIMO se ha desarrollado desde finales del siglo XX hasta hoy de la siguiente manera:

- **Antes de 1998,** se desarrollan e implementan detectores de detección óptima ML (Maximum Likelihood) basados en la búsqueda exhaustiva y detectores sub-óptimos lineales que son capaces de alcanzar elevadas tasas de transmisión a costa de penalizar la tasa de errores (BER).

  En el año 1997 se publica la norma IEEE 802.11, que especifica las normas de funcionamiento para las redes de área local inalámbrica (WLAN). En sus inicios la norma especifica velocidades de transmisión teóricas de 1 y 2 Mbps a través de infrarojos (IR).

- **En 1998,** los laboratorios Bell publican e implementan el decodificador lineal V-BLAST [2], basado en cancelación sucesiva de interferencias (SIC). Con él se consigue una notable mejora de las prestaciones BER, estando éstas todavía muy alejadas de la detección óptima.

- **En 1999,** E. Viterbo y J. Boutros publican una aplicación del algoritmo, no lineal, "Sphere Decoding" [3], basado en los trabajos de mediados de los 80 de los matemáticos M. Pohst y U. Fincke [4][5]. Este algoritmo alcanza la detección óptima pero como contrapartida su tasa de detección es variable.

  Paralelamente se aprueba la norma 802.11b que opera en la banda de 2.4 GHz y alcanza una velocidad de transmisión de 11 Mbps a través de multiplexación CSMA/CA.

- **En 2002,** se produce una eclosión de publicaciones de nuevos algoritmos y mejoras de los existentes para la detección MIMO:

  - K.W. Wong propone una arquitectura para la decodificación de MIMO, basada en el algoritmo KBest [6], tradicionalmente denominado "M algorithm" [7]. Este algoritmo se basa en "Sphere Decoding" y consigue una tasa de decodificación fija a costa de penalizar la tasa de errores (BER).

  - E. Agrell [8] publica una serie de mejoras significativas que incrementan la eficiencia de implementación del algoritmo "Sphere Decoding". Cabe destacar la introducción, en el algoritmo original de Pohst, de la técnica de enumeración de candidatos Schnorr-Euchner [9].
  - H. Yao [10] propone un mecanismo de preprocesado de la matriz de canal, "*Lattice Reduction*" (LR), que mejora notablemente la detección posterior mediante arquitecturas V-BLAST.

- **Entre 2003 y 2007,** aparecen múltiples arquitecturas hardware VLSI basadas en los algoritmos anteriormente citados y en mejoras propuestos de los mismos para la decodificación de esquemas 4x4 16-QAM . Destacan los trabajos de A. Burg [11][12][13] y Z. Guo [14][15]. A. Burg presenta en [13] una arquitectura basada en KBest cuya implementación en ASIC alcanza una tasa de 424 Mbps y Z. Guo presenta en [15] una primera arquitectura para decodificación "Soft-Output". Ésta permite el uso de decodificadores de corrección de errores "soft".

  En junio de 2003 se ratifica el estándar 802.11g, cuya velocidad teórica es de 54Mbps a 2.4 GHz y en enero de 2004 se forma el grupo de trabajo para desarrollar el 802.11n que sentará las bases de la comunicación MIMO.

- **En 2007,** T. Zhang desarrolla una arquitectura, "Relaxed KBest"[16], para detección "Soft-Output" de MIMO 4x4 64-QAM implementada en CMOS 0.13 μm. Para ello desarrolla un mecanismo de ordenación sub-óptimo del algoritmo KBest.

- **Desde 2008,** hasta la actualidad se desarrollan múltiples arquitecturas basadas en mejoras sobre algoritmos existentes y combinaciones de los mismos. Se presentan implementaciones VLSI en CMOS 0.13 μm de sistemas MIMO 4x4 64-QAM que alcanzan tasas del orden 800 Mbps. En los últimos trabajos [17] se incorpora la herramienta de pre-procesado (LR) en algoritmos KBest sobre sistemas MIMO 8x8 64-QAM para alcanzar tasas del orden de los 3Gbps.

1.3. Objetivos y contribuciones de la tesis

El objetivo de esta tesis es el desarrollo e implementación ASIC de una arquitectura de baja complejidad para la detección MIMO basada en el algoritmo de cancelación de interferencias denominado SPA ("*Successive Projection Algorithm*") [18][19]. Este algoritmo, sin implementación ASIC publicada hasta el momento, se engloba dentro de la familia de los detectores lineales de cancelación de interferencias (OSIC), caracterizados por

su baja complejidad computacional y como consecuencia de esto por una alta tasa de detección a costa de bajas prestaciones BER. Lo que diferencia a este algoritmo de otros pertenecientes a su misma familia es que sus autores definen un mecanismo para alcanzar de forma iterativa la detección óptima ML ("Maximum Likelihood"). Si bien este mecanismo ESPA ("Extended-SPA") puede alcanzar en el límite la detección óptima ML, su implementación hardware en estas condiciones es inviable debido la alta complejidad computacional necesaria para alcanzar estas prestaciones. Por lo tanto, el reto que se afronta en este trabajo es desarrollar una arquitectura de baja complejidad computacional que haga viable y competitiva, frente a otras arquitecturas lineales y no lineales, la implementación de este algoritmo.

Si bien la eficiencia espectral aumenta de forma notable con la incorporación de la detección MIMO, también lo hace con la utilización de esquemas de modulación elevados. Para este fin, el estándar de comunicación 3GPP(LTE) en su revisión 12 ha elevado la máxima modulación soportada desde 64-QAM hasta 256-QAM. El estado del arte de la detección MIMO descrito en la sección anterior presenta múltiples arquitecturas, en su mayoría basadas en el algoritmo no lineal KBest, que obtienen buenas prestaciones para decodificadores MIMO 4x4 64-QAM. Algunas de estas arquitecturas pueden extenderse para decodificar 256-QAM [20], pero no hay una valoración real ni del coste computacional asociado a esta extensión, ni de las prestaciones finales obtenidas.

Por lo tanto, el primer objetivo a cubrir es la realización de un estudio exhaustivo de los diferentes detectores, algoritmos y arquitecturas publicados en la literatura especializada para obtener mediante simulaciones, las prestaciones BER y la carga computacional alcanzadas por los mismos en diferentes escenarios de transmisión. Las conclusiones de este estudio ofrecen un primer marco para la toma de decisiones sobre la viabilidad de implementación del algoritmo SPA, objeto de estudio de esta tesis.

La tarea de desarrollar un detector, cuya implementación *hardware* sea viable y competitiva, tomando como punto de partida el algoritmo SPA original publicado, no es trivial. Los detectores basados en este algoritmo ofrecen como valor añadido, frente a otros detectores SIC, la posibilidad de generar de forma iterativa distintas soluciones. Sin embargo sus autores no establecen ningún método para controlar las repeticiones en las iteraciones, ni se estudia el número de iteraciones requeridas para aproximar la solución ML con un coste computacional razonable.

Como segundo objetivo, se aborda la búsqueda de los mecanismos de control de repeticiones que hagan viable la implementación *hardware* del algoritmo SPA. Para ello, tomando como punto de partida el algoritmo original, se realizará un estudio en profundidad del mismo y las propuestas algorítmicas necesarias para la obtención de las salidas *hard output* y *soft*

output. Mediante simulaciones se validará la viabilidad y competitividad, en cuanto a prestaciones BER/FER y carga computacional se refiere, de los detectores desarrollados frente a otros publicados en la literatura especializada.

Finalmente, como último objetivo, se diseñará una arquitectura VLSI, que maximice la tasa de detección y minimice el área de ocupación del *hardware*. Con ella, se implementarán cuatro detectores con diferentes prestaciones, salidas (*hard output* y *soft output*) y en dispositivos FPGA y ASIC. Se realizará un estudio de las topologías *hardware* que incrementan la tasa de detección y una comparación de los detectores desarrollados frente a los mejores publicados en la literatura especializada.

Como contribuciones de la tesis destacan las siguientes:

- Descripción detallada de los diferentes decodificadores MIMO, lineales y no lineales, utilizados en la literatura especializada y una comparación de prestaciones y complejidad numérica de los mismos.
- Desarrollo de los mecanismos de control de repeticiones *Simplied-ESPA* y *Table-ESPA* que hacen viable y competitiva la implementación del algoritmo SPA.
- Adaptación de los métodos de obtención de las salidas *hard output* y *soft output* al algoritmo SPA.
- Desarrollo de la primera arquitectura *hardware* para el algoritmo SPA.

En definitiva, en este trabajo se presenta un decodificador MIMO configurable, con capacidad de decodificar sistemas MIMO con diferentes configuraciones de antenas, desde 2x2 hasta 4x4, diferentes constelaciones, desde QPSK hasta 256-QAM y ofreciendo las salidas *hard output* y *soft output*. La implementación ASIC del detector ofrece resultados muy competitivos frente a otras arquitecturas flexibles publicadas a 64-QAM [21], e incorpora como novedad, además de la mayor configurabilidad, la decodificación de 256-QAM, sin incrementar la complejidad computacional.

1.4. Metodología

En esta sección se expone la metodología seguida para abordar los objetivos de esta tesis doctoral. En primer lugar se ha realizado una búsqueda bibliográfica para identificar el estado del arte y los algoritmos más relevantes que son usados como referencia. Posteriormente se han evaluado las prestaciones y se ha realizado un análisis de precisión finita de estos algoritmos. Para esto, se ha modelado un sistema de comunicaciones en Matlab, con el que se realizan simulaciones para evaluar las prestaciones siguiendo

el método de Montecarlo. El análisis de precisión finita se realiza también mediante simulación, por lo que ha sido necesaria la implementación de los algoritmos con precisión finita en Matlab. Para cada algoritmo se evalúan diferentes esquemas de cuantificación y se determina el que mejor compromiso entre número de bits y prestaciones consigue.

Para el análisis de los algoritmos propuestos se sigue la misma metodología que para el análisis de los algoritmos de referencia: en primer lugar se han implementado los modelos de punto flotante y precisión finita y a continuación se han simulado utilizando el entorno desarrollado en Matlab. El principal objetivo de los algoritmos propuestos es dotar de viabilidad de implementación *hardware* al algoritmo original SPA, manteniendo unas prestaciones similares a las de los algoritmos de referencia, y maximizando la configurabilidad en cuanto a parámetros del sistema MIMO se refiere.

Para la implementación *hardware* de los decodificadores se utiliza el lenguaje de descripción *hardware* VHDL (*Very High Speed Integrated Circuit Hardware Description Language*). Los modelos *hardware* primero se verifican funcionalmente a nivel de código, comparándolos con los modelos de precisión finita previamente desarrollados. Esta verificación se realiza con la herramienta Modelsim de Mentor Graphics, siguiendo el siguiente flujo de trabajo:

1. Se realiza una simulación con el modelo de precisión finita y se almacenan tanto los datos de entrada como los datos de salida del decodificador (algoritmo de decodificación).

2. En un banco de pruebas, codificado en VHDL, se leen los datos de entrada, se simula el modelo *hardware* y se comparan los datos generados por éste con los obtenidos con el modelo *software* de precisión finita.

3. Si las comparaciones son válidas (los datos son exactamente iguales) se da como verificado el modelo *hardware*.

Una vez validado el modelo *hardware*, éste es implementado en un ASIC o en un dispositivo FPGA. Las implementaciones en un ASIC se realizan usando la librería de celdas estándar Faraday de 90nm con 8 capas metálicas [22]. Para la síntesis se ha empleado la herramienta de Cadence RTL *compiler* y para el emplazado y rutado la herramienta SOC *encounter* de Cadence, la cual integra la herramienta para el análisis de tiempos. Las implementaciones en FPGA se han realizado con la herramienta ISE de Xilinx, las cuales integran sus respectivas herramientas de análisis de tiempo. Los modelos post emplazado y rutado generados por las diferentes herramientas se verifican funcionalmente siguiendo el mismo flujo que para la verificación a nivel de código.

El flujo de trabajo que se sigue para la implementación en un ASIC se describe a continuación:

1. Se realiza una primera síntesis sin constante de tiempo.
2. Se realiza una segunda síntesis ajustando la constante de tiempo al 80 % del camino crítico obtenido en la síntesis anterior.
3. A continuación se realiza el emplazado y rutado. Este se hace de manera iterativa, ajustando el porcentaje de área requerida hasta obtener una mínima penalización del camino crítico (comparado con el camino crítico de la síntesis). Si el porcentaje de área es inferior al 40 % o la penalización del camino crítico es muy alta (superior al 50 %) se incrementa la constante de tiempo (de la sintesis) y se vuelve al paso 2.

La implementación en un dispositivo FPGA también se realiza de manera iterativa, con el fin de ajustar la frecuencia de trabajo a la máxima posible (mínimo camino critico). Para esto se sigue el siguiente flujo de trabajo.

1. Se realiza una primera compilación con una constante de tiempo alta.
2. Con el resultado de camino crítico, obtenido con la herramienta de análisis de tiempo, se ajusta la constante de tiempo y se realiza de nuevo la compilación.
3. El paso 2 se repite hasta conseguir el mínimo camino crítico sin que se produzcan violaciones de tiempo.

1.5. Estructura de la tesis

Como punto de partida para el desarrollo de esta tesis en el capítulo 2 se describe el modelo matemático del sistema MIMO junto con los modelos de detección utilizados. Se realiza, además, una detallada descripción de los diferentes decodificadores MIMO, lineales y no lineales, utilizados en la literatura especializada y una comparación de prestaciones y complejidad numérica de los mismos.

El capítulo 3 presenta los fundamentos teóricos del Algoritmo de Proyecciones Sucesivas (SPA), motivo fundamental de esta tesis, y se exponen las modificaciones algorítmicas necesarias para su posterior implementación, así como la adaptación de los métodos para la obtención de las salidas *hard* y *soft output* al algoritmo SPA. Se realiza, además, una comparativa de prestaciones y complejidad computacional con el detector más competitivo KBest.

En el capítulo 4 se realiza una descripción detallada de la arquitectura propuesta para la implementación en FPGA y VLSI del decodificador *hard output* y *soft output* SPA, en sus versiones simplificada y extendida, y se presentan los resultados de implementación en ambos dispositivos. Además, se realiza un estudio topológico para incrementar la tasa de detección y se presenta una tabla comparativa de resultados de implementación con otras arquitecturas publicadas.

Por último, en el capítulo 5 se exponen las conclusiones de este trabajo.

En el anexo A se muestran los esquemas de trabajo y el cronograma de señales de control utilizados para el desarrollo del trabajo, así como la tabla de precisiones de los datos ajustadas para minimizar las pérdidas de implementación.

Capítulo 2

MIMO: sistema y algoritmos de decodificación

En el estudio de la detección de las señales transmitidas sobre canales lineales MIMO hay una serie de conocimientos previos y herramientas clave fundamentales para la comprensión del mismo. Los objetivos que cubre este capítulo se resumen en los siguientes puntos:

- Descripción del sistema MIMO bajo consideración e introducción de la notación matemática utilizada en este trabajo.
- Descripción de los modelos y herramientas matemáticas útiles para la detección MIMO.
- Enumeración, descripción y comparación en cuanto a prestaciones y complejidad de diferentes detectores MIMO.

En la sección 2.1 se presenta el modelo del sistema MIMO y la notación matemática utilizada. En la sección 2.2 se introducen los conceptos generales sobre la detección de señales MIMO. Las transformaciones matriciales ampliamente utilizadas por diferentes detectores MIMO se exponen en la sección 2.3, junto con una comparativa de prestaciones y complejidad de los mismos.

2.1. Modelo del sistema MIMO

Diferentes sistemas de comunicación de banda ancha combinan distintas técnicas de diversidad para aumentar la capacidad del canal y obtener así el mejor rendimiento. Estándares como el 802.11n combinan la tecnología MIMO con la modulación OFDM. Sin embargo, el objetivo de esta tesis se centra exclusivamente en la diversidad espacial que provoca la tecnología MIMO y por ende, en los beneficios asociados a ella. Sobre este escenario, el sistema en consideración se muestra en la figura 2.1, donde el número de antenas transmisoras es M_t y el número de antenas receptoras es N_r. Puesto que el foco de esta tesis se centra sobre la multiplexación espacial se asume siempre que $N_r \geq M_t$.

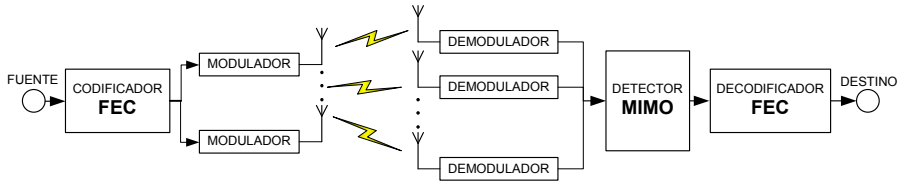


Figura 2.1: Esquema general de un sistema de comunicaciones MIMO

Transmisor: Para un sistema MIMO con M_t antenas transmisoras la señal transmitida será un vector $\mathbf{s_c}$ $(M_t \times 1)$. Las componentes de este vector son los símbolos transmitidos por cada antena. Estos símbolos se obtienen independientemente de la constelación compleja $\mathcal{X}^2$ perteneciente a la modulación digital q-QAM, donde q $= 2^B$, siendo B el número de bits por símbolo. Con todo esto, la tasa alcanzada por un sistema MIMO con M_t antenas transmisoras operando con multiplexación espacial se expresa como $R = M_t B$ bits por canal (bpcu). Para eliminar la influencia del número de antenas y del esquema de modulación el vector $\mathbf{s_c}$ se ha normalizado antes de la transmisión, de forma que la potencia media transmitida sea la unidad $(E\|\mathbf{s_c}\|^2 = 1)$.

Canal MIMO: La figura 2.2 muestra con detalle los agentes involucrados en el modelo equivalente banda-base del canal MIMO, cuya relación de entrada-salida se expresa mediante la siguiente ecuación:

$$\mathbf{y}_c = \mathbf{H_c}\mathbf{s_c} + \mathbf{n_c}. \tag{2.1}$$

La matriz compleja $\mathbf{H_c}$ de dimensiones $N_r \times M_t$ modela un canal Rayleigh "flat-fading"no selectivo en frecuencia y quasi-estático. Sus componentes h_{ij} son variables Gaussianas complejas de media cero y varianza $\sigma^2 = 1$ por dimensión compleja, y representan la función

de transferencia entre la antena transmisora j y la antena receptora i. El vector N_r-dimensional n_c modela el rruido blanco Gaussiano, independiente e identicamente distribuido por dimensión compleja, de media cero y varianza N_0. La SNR se define como la relación entre la potencia total transmitida, normalizada a la unidad, y la varianza del ruido térmico, de acuerdo con la siguiente expresión:

$$SNR = \frac{1}{N_0} \tag{2.2}$$

Receptor: Las antenas del receptor captan las N_r componentes del vector recibido $\mathbf{y_c}$. Teniendo en cuenta que tanto la potencia transmitida como la varianza del canal están normalizadas a la unidad, la relación señal a ruido media recibida (sobre diversas realizaciones de canal) por antena receptora será directamente la SNR de 2.2.
La tarea del detector MIMO es obtener la mejor estimación $\widehat{\mathbf{s}}_\mathbf{c}$ posible del vector transmitido $\mathbf{s_c}$ basada en el vector recibido $\mathbf{y_c}$. Para ello se asume en esta tesis que el receptor cuenta con una estimación $\widehat{\mathbf{H}}_\mathbf{c}$ de la matriz de canal $\mathbf{H_c}$.

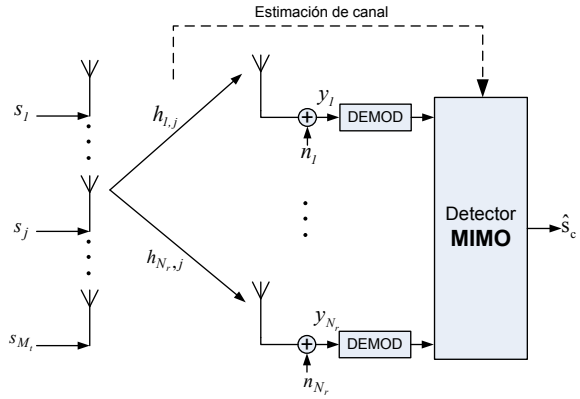


Figura 2.2: Esquema de la detección de señales MIMO

2.2. Detección de señales MIMO

Subyacente a la implementación de los sistema de comunicaciones MIMO, existen determinados aspectos generales a tener en cuenta. En particular, de la observación de diferentes detectores MIMO se desprende como característica común que pueden ser divididos en dos fases como ilustra la figura 2.3.

- **Fase de prepocesado** o procesado de canal. En esta fase se realizan todas las operaciones necesarias, fundamentalmente sobre la matriz de canal, solamente cuando hay un cambio en las condiciones del canal. En la sección 2.2.2 se muestran dos transformaciones de la matriz de canal ampliamente utilizadas en detectores MIMO descritos en la literatura especializada:

  - *La transformación QR* no solo como herramienta de triangularización sino además como paso previo para la inversión de la matriz de canal.
  - *La descomposición en valores reales (RVD)* que transforma el sistema MIMO descrito en el plano complejo en la ecuación 2.1, a un problema de detección de símbolos reales.

- **Fase de detección de símbolos**. Esta fase comprende todas aquellas operaciones necesarias para realizar la estimación de los símbolos transmitidos a partir del vector recibido. La sección 2.2.1 describe dos modelos de detectores MIMO según el objetivo que persigan:

  - *La detección hard output* proporciona la estimación "dura" de cada bit, es decir, se toma una decisión sobre si el bit recibido es un 1 o un 0.
  - *La detección soft output* proporciona la probabilidad a posteriori (APP) de cada bit. Esta información es utilizada por un sistema de corrección de errores (FEC, Forward Error Correction).

  La figura 2.1 muestra el esquema general de detección para ambos modelos. Aunque el objetivo común es obtener la estimación $\widehat{\mathbf{s}}_\mathbf{c}$ del vector transmitido $\mathbf{s_c}$ a partir del vector recibido $\mathbf{y_c}$, hay que hacer notar que el uso que hacen los sistemas FEC de la información APP mejoran notablemente las prestaciones alcanzadas por la detección *soft output*.

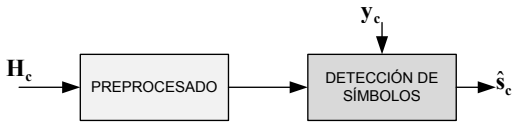


Figura 2.3: Diagrama de bloques de las fases de la detección de señales MIMO.

En la práctica, se asume un canal invariante sobre un gran número de símbolos recibidos, por lo tanto la fase de preprocesado es menos crítica. Sin embargo, en escenarios de alta movilidad, con limitaciones de latencia muy

estrictas o en sistemas MIMO de banda ancha, esta hipótesis de trabajo puede no ser cierta. En cualquier caso, la consideración de tratar las dos fases por separado está justificada, ya que son de naturaleza muy diferente tanto en lo que se refiere a requerimientos de diseño como a prestaciones.

2.2.1. Modelos de detección MIMO

Según el objetivo perseguido por la detección, la literatura especializada distingue dos modelos de detección MIMO: la detección *hard output* y la detección *soft output*.

1. **La detección *hard output* (ML)** tiene como objetivo proporcionar la estimación "hard"de cada bit. Para un sistema MIMO la detección óptima ML (Maximum Likelihood) consiste en encontrar un punto $\widehat{\mathbf{s}}_c$ del "*lattice*", perteneciente al espacio $(\mathcal{X}^2)^{M_t}$, que minimice la distancia euclidea entre el punto transformado $\mathbf{H_c}\widehat{\mathbf{s}}_\mathbf{c}$ y el vector recibido $\mathbf{y_c}$, según expresa la siguiente ecuación:

$$\hat{\mathbf{s}}_\mathbf{c} = \arg \min_{\mathbf{s_c} \in (\mathcal{X}^2)^{M_t}} d(\mathbf{s_c}) \quad \text{con} \quad d(\mathbf{s_c}) = \|\mathbf{y_c} - \mathbf{H_c}\mathbf{s_c}\|^2, \tag{2.3}$$

   donde $\mathcal{X}^2$ es el conjunto de los q valores complejos de la constelación.

2. **La detección *soft output* (LLR)** tiene como objetivo encontrar la información APP de cada bit. Esta información se expresa usualmente como un valor "Log-Likelihood Ratio"(LLR). Para M_t símbolos transmitidos de B bits cada uno, se define el LLR del bit b_{ij} como:

$$L\left(b_{ij}|\mathbf{y_c}\right) = \ln\left(\frac{\Pr\left(b_{ij}=+1|\mathbf{y_c}\right)}{\Pr\left(b_{ij}=-1|\mathbf{y_c}\right)}\right), \tag{2.4}$$

   donde b_{ij} representa el j-ésimo bit del i-ésimo símbolo. Debido a la gran dificultad de computación de la ecuación (2.4), se suele adoptar como simplificación estándar para el cálculo del LLR la siguiente ecuación [23][24]:

$$L\left(b_{ij}|\mathbf{y_c}\right) \approx \left(\min_{\mathbf{s_c} \in \mathcal{X}_{ij}^{(-1)}} \Lambda - \min_{\mathbf{s_c} \in \mathcal{X}_{ij}^{(1)}} \Lambda\right), \tag{2.5}$$

   donde $\Lambda = \dfrac{1}{N_0}\|\mathbf{y_c} - \mathbf{H_c}\mathbf{s_c}\|^2$ y los conjuntos $\mathcal{X}_{ij}^{(1)}$ y $\mathcal{X}_{ij}^{(-1)}$ incluyen todos los vectores de símbolos cuyo j-ésimo bit del i-ésimo símbolo son 1 y -1 respectivamente.

2.2.2. Herramientas de preprocesado

En esta fase se realizan todas las operaciones necesarias, fundamentalmente sobre la matriz de canal, solamente cuando hay un cambio en las condiciones del canal. A continuación se describen dos operaciones básicas para el tratamiento de las señales MIMO antes de su detección. Estas operaciones se aplican sobre la ecuación 2.1 que describe el modelo de un sistema MIMO, y sobre las ecuaciones 2.3 y 2.5 que describen la detección *hard output* ML y *soft output* LLR, respectivamente.

1. **Descomposición RVD**

   a) *Sistema MIMO:* La ecuación 2.1 describe un sistema MIMO en banda base complejo, es decir tanto los símbolos de la constelación $\mathcal{X}^2$ asociada a la modulación digital q-QAM, como la matriz de canal, son valores complejos. Sin embargo, para constelaciones q-QAM rectangulares, es posible descomponer el modelo de señal complejo M_t-dimensional expresado en la ecuación 2.1 en un problema real $2M_t$-dimensional de la siguiente manera:

   $$\begin{bmatrix} Re\{\mathbf{y_c}\} \\ Im\{\mathbf{y_c}\} \end{bmatrix} = \begin{bmatrix} Re\{\mathbf{H_c}\} & -Im\{\mathbf{H_c}\} \\ Im\{\mathbf{H_c}\} & Re\{\mathbf{H_c}\} \end{bmatrix} \begin{bmatrix} Re\{\mathbf{s_c}\} \\ Im\{\mathbf{s_c}\} \end{bmatrix} + \begin{bmatrix} Re\{\mathbf{n_c}\} \\ Im\{\mathbf{n_c}\} \end{bmatrix}. \tag{2.6}$$

   La descripción del sistema MIMO con valores reales se expresa ahora mediante la siguiente ecuación:

   $$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n}, \tag{2.7}$$

   donde la matriz de canal real es $\mathbf{H} \in \mathbb{R}^{N \times M}$ ($N = 2N_r$ y $M = 2M_t$), $\mathbf{y} \in \mathbb{R}^N$ y $\mathbf{s} \in \mathcal{X}^M$, siendo ahora $\mathcal{X}$ el conjunto de los $\mathcal{B}$ valores reales de la constelación, e.g., $\mathcal{X} = \{-3, -1, 1, 3\}$ en el caso de 16-QAM.

   b) *Detección hard output (ML):* Considerando la transformación del sistema expresada por la ecuación anterior, la detección ML representada por la ecuación 2.3 puede reformularse ahora como:

   $$\hat{\mathbf{s}} = \arg \min_{\mathbf{s} \in (\mathcal{X})^M} d(\mathbf{s}) \quad \text{con} \quad d(\mathbf{s}) = \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2, \tag{2.8}$$

   donde $\mathcal{X}$ es el conjunto de los $\mathcal{B}$ valores reales de la constelación.

   c) *Detección soft output (LLR):* Para el cálculo del LLR de cada bit la ecuación 2.5 sigue siendo válida tomando ahora:

   $$\Lambda = \frac{1}{N_0} \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2. \tag{2.9}$$

2. **Descomposición QR**

a) *Sistema MIMO:* El modelo del sistema MIMO representado por la ecuación 2.1 puede re-escribirse de la siguiente manera:

$$\mathbf{y_c} = \mathbf{QRs_c} + \mathbf{n_c},$$

donde la matriz de canal $\mathbf{H_c}$ se ha triangularizado a través de la descomposición QR[1] ($\mathbf{H_c} = \mathbf{QR}$), siendo $\mathbf{Q}$ una matriz unitaria de dimensiones $N_r \times N_r$ y $\mathbf{R}$ una matriz triangular superior de dimensiones $N_r \times M_t$.
Premultiplicacando la ecuación anterior por $\mathbf{Q^T}$ se obtiene:

$$\mathbf{Q^T y_c} = \mathbf{Rs_c} + \mathbf{Q^T n_c}$$

$$\widehat{\mathbf{y}}_\mathbf{c} = \mathbf{Rs_c} + \mathbf{Q^T n_c} \quad \text{con} \quad \widehat{\mathbf{y}}_\mathbf{c} = \mathbf{Q^T y_c}, \tag{2.10}$$

donde a través de la rotación del vector $\mathbf{y_c}$ con la matriz $\mathbf{Q^T}$ ahora el sistema MIMO se ha transformado en un problema triangular, facilitándose así las operaciones de los detectores basados en algoritmos de busqueda en árbol y la inversión de la matriz de canal, ampliamente utilizada en los algoritmos de detección lineal y SIC descritos en la sección 2.3.

b) *Detección hard output (ML):* A la vista de la ecuación anterior, la detección ML representada por la ecuación 2.3 puede re-escribirse ahora como:

$$\begin{aligned} &\hat{\mathbf{s}}_\mathbf{c} = \arg\min_{\mathbf{s_c} \in (\mathcal{X}^2)^{M_t}} d(\mathbf{s_c}) \\ &d(\mathbf{s_c}) = \|\widehat{\mathbf{y}}_\mathbf{c} - \mathbf{Rs_c}\|^2 \quad \text{con} \quad \widehat{\mathbf{y}}_\mathbf{c} = \mathbf{Q^T y_c}, \end{aligned} \tag{2.11}$$

En el caso general de un sistema MIMO $N_r \times M_t$, resulta muy útil reducir este problema de minimización a las dimensiones $M_t \times M_t$, para ello consideremos la representación matricial de la expresión $\mathbf{Q^T y_c} - \mathbf{Rs_c}$:

$$\begin{bmatrix} & & \\ & \mathbf{Q_1^T} & \\ & & \\ - & - & - \\ & & \\ & \mathbf{Q_2^T} & \\ & & \end{bmatrix} \begin{bmatrix} y_1 \\ \\ \vdots \\ y_{M_t} \\ \\ \vdots \\ y_{N_r} \end{bmatrix} - \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1M_t} \\ 0 & r_{22} & & r_{2M_t} \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & r_{M_tM_t} \\ 0 & 0 & \dots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} s_1 \\ \\ \vdots \\ \\ s_{M_t} \end{bmatrix},$$

[1] Se requiere que los valores de la diagonal de $\mathbf{R}$ sean reales y positivos.

donde la matriz cuadrada $\mathbf{Q^T}$ de dimensiones $(N_r \times N_r)$ se ha dividido en dos sub-matrices: $\mathbf{Q_1^T}$ de dimensiones $(M_t \times N_r)$ y $\mathbf{Q_2^T}$ de dimensiones $(N_r - M_t \times N_r)$. El lector puede abservar además, que la matriz $\mathbf{R}$ tiene ceros desde la fila $M_t + 1$ hasta al final. Este hecho hace que la distancia $d(\mathbf{s_c})$ de la ecuación 2.11 pueda ser expresada como:

$$d(\mathbf{s_c}) = \|\mathbf{Q_1^T y_c} + \mathbf{Q_2^T y_c} - \mathbf{R s_c}\|^2 = \|\mathbf{Q_1^T y_c} - \mathbf{R s_c}\|^2 + cte^2, \tag{2.12}$$

por lo tanto la ecuación 2.11 puede ser reformulada eliminando el término constante (que no depende de los símbolos $\mathbf{s_c}$) de la siguiente manera:

$$\begin{aligned} &\hat{\mathbf{s}}_\mathbf{c} = \arg \min_{\mathbf{s_c} \in (\mathcal{X}^2)^{M_t}} d(\mathbf{s_c}) \\ &d(\mathbf{s_c}) = \|\widehat{\mathbf{y}}_\mathbf{c} - \mathbf{R s_c}\|^2 \quad \text{con} \quad \widehat{\mathbf{y}}_\mathbf{c} = \mathbf{Q_1^T y_c}, \end{aligned} \tag{2.13}$$

donde ahora la matriz R es triangular superior de dimensiones $M_t \times M_t$.

c) *Detección soft-output (LLR):* Para el cálculo del LLR de cada bit la ecuación 2.9 sigue siendo válida tomando ahora:

$$\Lambda = \frac{1}{N_0} \|\widehat{\mathbf{y}}_\mathbf{c} - \mathbf{R s_c}\|^2 \quad \text{con} \quad \widehat{\mathbf{y}}_\mathbf{c} = \mathbf{Q_1^T y_c}, \tag{2.14}$$

3. **Algoritmos "*Lattice Reduction*" (LR)[10],** es una técnica de preprocesado de la matriz de canal que mejora significativamente las prestaciones de BER de los detectores MIMO. La aplicación del método LR reduce la probabilidad de error, provocada por las perturbaciones de ruido, en la detección de símbolos. Esto se consigue al transformar la matriz del sistema en una matriz cuasi ortogonal. Hay muchos algoritmos LR en la literatura matemática, sin embargo en términos de su aplicación en la detección MIMO, hay tres fundamentales:

   a) *El algoritmo de Lenstra, Lenstra y Lovász (LLL)[25]* es el más utilizado, sin embargo su implementación es de alta complejidad computacional y de tiempo de ejecución no determinístico. Es un algoritmo iterativo, lo que provoca tasas de detección variables en función de la ortogonalidad de la matriz de entrada al sistema y un elevado número de ciclos de de ejecución.

   b) *El algoritmo de Seysen[26]* es tambien iterativo y aunque consigue mejores prestaciones BER que el LLL, su carga computacional es superior a éste en cada iteración, sin embargo, requiere de menos iteraciones para alcanzar la ortogonalidad de la matriz de canal[27].

c) *El algoritmo de Brun[28]* en comparación con los dos anteriores, es el que presenta menor complejidad computacional a costa de peores prestaciones BER.

En general el algoritmo LLL consigue el equilibrio entre complejidad computacional y prestaciones BER. En la literatura reciente [29][17] se ha propuesto la implementación VLSI de esta técnica de preprocesado combinada con el algoritmo de detección KBest (2.3.3.2), con una notable reducción de la complejidad.

2.3. Algoritmos de detección MIMO

Las técnicas de preprocesado descritas en la sección anterior operan sobre la matriz de canal $\mathbf{H_c}$ y sobre el vector de símbolos recibidos $\mathbf{y_c}$ para, sobre éstos, aplicar diferentes técnicas de detección basadas en diferentes algoritmos matemáticos. El objetivo final es encontrar las soluciones *hard output* y/o *soft output*.

En esta sección se describen los algoritmos utilizados por diferentes detectores MIMO en la literatura especializada. Estos detectores se agrupan fundamentalmente en tres familias:

- *Detectores basados en búsqueda exhaustiva*, que computan todos los candidatos posibles y seleccionan el de mayor probabilidad según la ecuación 2.3.

- *Detectores lineales y de cancelación sucesiva de interferencias (SIC)*, que contrarrestan el efecto del canal premultiplicando el vector recibido por una matriz estimadora.

- *Detectores no lineales o de búsqueda en árbol*, basados en la triangularización de la matriz de canal y en la búsqueda recursiva de candidatos en un árbol de símbolos, específico para cada sistema MIMO $N_r \times M_t$ $q-QAM$.

Además de la descripción de los diferentes detectores pertenecientes a las familias citadas, en la sección 2.3.4 se presenta una comparación detallada entre ellos basada en dos parámetros de diseño: complejidad computacional y prestaciones BER alcanzadas. El balance entre estos dos parámetros de diseño constituye un reto de cara a una futura implementación VLSI de los mismos.

2.3.1. Búsqueda exhaustiva

Dado el vector $\mathbf{y_c}$, la detección ML descrita por la ecuación 2.3 encuentra el símbolo $\widehat{\mathbf{s}}_\mathbf{c} \in (\mathcal{X}^2)^{M_t}$ que con mayor probabilidad ha sido transmitido:

$$\hat{\mathbf{s}}_\mathbf{c} = \arg \min_{\mathbf{s_c} \in (\mathcal{X}^2)^{M_t}} \|\mathbf{y_c} - \mathbf{H_c s_c}\|^2.$$

Una implementación directa de esta ecuación mediante búsqueda exhaustiva sobre la totalidad del conjunto de posibles vectores $(\mathcal{X}^2)^{M_t}$, resuelve el problema y encontraría $\widehat{\mathbf{s}}_\mathbf{c}$. Sin embargo, el precio que hay que pagar por realizar este tipo de implementación y obtener las mejores prestaciones en cuanto a BER y a diversidad espacial se refiere, es demasiado alto. La complejidad del detector crece exponencialmente con la tasa R ($2^R = 2^{M_t B}$). Para ilustrar la severidad de este crecimiento sirva como ejemplo un sistema MIMO 4x4 con las modulaciones QPSK, 16-QAM y 64-QAM (R=8, 16 y 24 bpcu). El número de combinaciones posibles de símbolos candidatos a ser solución de la ecuación ML es 256, 65536 y 16777216, respectivamente.

Un aspecto positivo de este detector es que puede generar una lista con los mejores candidatos, los que se acercan más a la solución óptima, que puede ser utilizada para obtener la salida *soft output*.

2.3.2. Detección lineal y SIC

2.3.2.1. Detección lineal

Los métodos basados en la detección lineal consideran la relación de entrada-salida de un sistema MIMO (ecuación 2.1) como una estimación lineal sin restricciones del problema, cuya solución es alcanzada a través de mínimos cuadrados, *zero-forzing* (ZF), o del cálculo del error cuadrático medio (MMSE). Corresponde, por lo tanto al receptor, intentar contrarrestar el efecto del canal premultiplicando el vector recibido por una matriz estimadora $\mathbf{G_c}$ para obtener

$$\hat{\mathbf{x}}_\mathbf{c} = \mathbf{G_c y_c}. \tag{2.15}$$

La ecuación 2.15 devuelve una estimación ($\hat{\mathbf{x}}_\mathbf{c} \in \mathbb{C}^{M_t}$) sin restricciones del vector transmitido $\mathbf{s_c}$, lo que ignora por completo el hecho de que los valores de $\mathbf{s_c}$ están restringidos al conjunto limitado de los puntos de la constelación $\mathcal{X}^2$. Por lo tanto, el proceso de detección requiere de un paso adicional, mapear a puntos válidos de la constelación cada componente del vector $\hat{\mathbf{x}}_\mathbf{c}$ de la siguiente manera:

$$\hat{s}_i = Q(\hat{x}_i), \tag{2.16}$$

donde $Q(\cdot)$ es el operador "*slice*" (decisor) para un esquema de modulación determinado. La complejidad de este detector es básicamente el producto

matriz vector de la ecuación 2.15 y el operador *slice* (unas cuantas sumas y comparaciones). Sin duda, de todos los detectores expuestos en este trabajo es el que más baja complejidad computacional presenta, lo cual está en concordancia con las bajas prestaciones de BER alcanzadas.

Para realizar la estimación lineal se utilizan fundamentalmente dos estimadores: el estimador zero-forcing (ZF) y el basado en el cálculo del error cuadrático edio (MMSE)[30].

1. **Estimación *zero-forcing* (ZF):** El estimador de mínimos cuadrados $\mathbf{G_c}$ se obtiene a través del cálculo de la pseudoinversa de Moore-Penrose de la matriz $\mathbf{H_c}$ como:

$$\mathbf{G_c} = (\mathbf{H_c^H H_c})^{-1}\mathbf{H_c^H} = \mathbf{H_c}^{\dagger}. \tag{2.17}$$

   En el caso particular en el que $M_t = N_r$ la pseudoinversa de Moore-Penrose es directamente la inversa de $\mathbf{H_c}$, es decir, $\mathbf{G_c} = \mathbf{H_c^{-1}}$.
   De la aplicación de la pseudoinversa o en su caso la inversa a la ecuación 2.15 se obtiene:

$$\hat{\mathbf{x}}_\mathbf{c} = \mathbf{s} + \tilde{\mathbf{n}}_\mathbf{ZF} \quad \text{con} \quad \tilde{\mathbf{n}}_\mathbf{ZF} = \mathbf{G_c n_c}.$$

   De esta manera el canal efectivo entre el transmisor y el *slicer* del receptor es ahora la matriz identidad $\mathbf{I}$, eliminandose así la interferencia del canal en el flujo de símbolos. Por contra, esta separación tan efectiva de los datos transmitidos que produce la detección ZF puede tener como desventaja la enfatización del ruido aditivo, siendo ahora $\tilde{\mathbf{n}}_\mathbf{ZF}$.

2. **Estimación MMSE:** En lugar de anular los términos interferentes a costa del ruido, la detección MMSE minimiza el error esperado teniendo en cuenta el ruido. Se puede demostrar que el equilibrio óptimo entre cancelación de interferencias y enfatización de ruido se alcanza a través de la siguiente estimación:

$$\mathbf{G_c} = (\mathbf{H_c^H H_c} + \mathbf{M_t N_0 I})^{-1}\mathbf{H_c^H}, \tag{2.18}$$

   que tras ser sustituida en la ecuación 2.15 se obtiene:

$$\hat{\mathbf{x}}_\mathbf{c} = \tilde{\mathbf{H}}_\mathbf{c}\mathbf{s} + \tilde{\mathbf{n}}_\mathbf{MMSE} \quad \text{con} \quad \tilde{\mathbf{n}}_\mathbf{MMSE} = \mathbf{G_c n_c},$$

   donde ahora $\tilde{\mathbf{H}}_\mathbf{c} = \mathbf{G_c H_c}$ es el canal efectivo después de la ecualización MMSE.

2.3.2.2. Detección SIC

Los detectores SIC realizan la estimación del vector símbolo a símbolo, utilizando para ello las técnicas lineales vistas en la sección anterior. Una vez el símbolo ha sido detectado, su contribución es substraída del vector recibido. Como consecuencia, los errores en la detección de los primeros símbolos influyen negativamente en la detección de los siguientes, provocando finalmente un notable deterioro en las prestaciones BER del sistema. Para paliar este defecto se introducen técnicas de ordenación, con ellas se busca minimizar el error de detección en las sucesivas iteraciones. El algoritmo V-BLAST encuentra la ordenación que optimiza las prestaciones de BER del detector teniendo en cuenta exclusivamente la naturaleza del canal. Otros algoritmos de ordenación, por ejemplo la ordenación por norma de columnas, presentan una complejidad más reducida que el V-BLAST pero, sin embargo, las prestaciones BER alcanzadas son inferiores. Considerando el sistema MIMO representado por la ecuación 2.1, y siendo $\mathcal{I}^{(1)}$ el conjunto de índices $\{1, 2, \ldots, M_t\}$ que corresponden al orden natural de detección, en general un detector SIC con ordenación realizará en el nivel k las siguientes tareas:

Pseudocódigo 2.1 Algoritmo SIC

1. Aplica un criterio para seleccionar el índice i_k, entre los índices no seleccionados anteriormente,$i_k \in \mathcal{I}^{(k)}$.
2. Realiza la estimación de s_{i_k}, $\widehat{x}_{i_k} = \mathbf{g}_{i_k}^{(k)} \cdot \mathbf{y_c}^{(k)}$ donde $\mathbf{g}_{i_k}^{(k)}$ es la fila i_k de la matriz $\mathbf{G_c}^{(k)}$, estimador lineal (ZF ó MMSE) de la matriz $\mathbf{H_c}^{(k)}$.
3. Detecta el símbolo $\hat{s}_{i_k} = Q(\widehat{x}_{i_k})$ de la constelación, $s_{i_k} \in \mathcal{X}^2$, a través del operador *slice*, $Q(\cdot)$.
4. Realiza la cancelación del símbolo detectado sobre el vector de símbolos recibidos $\mathbf{y}_c^{(k+1)} = \mathbf{y}_c^{(k)} - \hat{s}_{i_k}\mathbf{h}_{i_k}^{(k)}$, siendo $\mathbf{h}_{i_k}^{(k)}$ la columna i_k de $\mathbf{H_c}^{(k)}$.
5. Anula la columna i_k de $\mathbf{H_c}^{(k)}$ para obtener $\mathbf{H_c}^{(k+1)}$ de la siguiente iteración, donde $\mathbf{H_c}^{(k+1)} = [\mathbf{h}_1^{(k)}, \ldots, \mathbf{h}_{i_k}^{(k)}, \ldots, \mathbf{h}_{M_t}^{(k)}]$, siendo $\mathbf{h}_{i_k}^{(k)} = [0]_{N_r \times 1}$.
6. Computa la nueva matriz estimadora $\mathbf{G_c}^{(k+1)}$ para la siguiente iteración, a partir de la matriz $\mathbf{H_c}^{(k+1)}$, utilizando ZF ó MMSE.
7. Elimina el índice detectado i_k del conjunto de índices, $\mathcal{I}^{(k+1)} = \mathcal{I}^{(k)} \backslash i_k$.

La ejecución de este algoritmo M_t veces dará como resultado una estimación sub-óptima (con prestaciones inferiores a las del detector ML de 2.3) $\hat{\mathbf{s_c}}$ del vector transmitido $\mathbf{s_c}$. Diferentes detectores pueden ser realizados aplicando cambios en los pasos 1 y 6 del algoritmo SIC, el cual es igualmente aplicable para la representación real del sistema MIMO expresada por la ecuación 2.7. Por ejemplo, en el caso de V-BLAST-ZF el criterio de ordenación aplicado en 1 sería $i_k = \arg\min_{j \in \mathcal{I}^{(k)}} \|\mathbf{G}_j^{(k)}\|^2$, y la matriz estimadora ZF del paso 6 se obtendría directamente de la ecuación 2.17.

Si el criterio de ordenación empleado en el paso 1 del algoritmo general depende exclusivamente de la naturaleza del canal, como sucede en el caso

citado de V-BLAST, el proceso de detección podría optimizarse separándolo en dos fases: fase de preprocesado y fase de detección.

1. **Fase de pre-procesado**: se obtiene el orden en el que se va a realizar la detección de los símbolos en la fase de detección. Para ello se ejecuta el algoritmo general M_t veces eliminando los pasos 2, 3 y 4 para encontrar:
   - un vector de permutación $\mathbf{P}$ sobre el orden natural de detección $\{1, 2, \ldots, M_t\}$,
   - la matriz estimadora $\mathbf{G_c}$ cuyas filas son los vectores de anulación necesarios para realizar la detección lineal con el orden de $\mathbf{P}$,
   - la matriz $\mathbf{H_c}$ con sus columnas permutadas según el orden establecido por $\mathbf{P}$.
2. **Fase de detección**: se ejecuta el algoritmo general M_t veces eliminando los pasos 1, 5, 6 y 7, y se le aplica al vector detectado la permutación $\mathbf{P}$ para así recuperar el orden inicial.

De esta manera, la fase de detección se descarga de una tarea tan compleja, desde el punto de vista computacional, como es el cálculo de la matriz estimadora $\mathbf{G_c}$, para el cual es necesario calcular la matriz pseudo-inversa de $\mathbf{H_c}$, consiguiendo así elevadas tasas de detección. En general las prestaciones alcanzadas por los detectores SIC varían en función de los criterios de ordenación y de las matrices estimadoras seleccionadas. El más óptimo, VBLAST, con una moderada carga computacional, mejora notablemente las prestaciones de los detectores lineales, quedando aún lejos de la curva de detección óptima ML.

2.3.2.3. Detección SPA

En [2] se demuestra que V-BLAST encuentra la ordenación óptima cuando únicamente interviene la matriz de canal, pero el algoritmo general SIC expuesto en la sección 2.3.2 posibilita la aplicación de técnicas de ordenación dependientes no sólo de la matriz de canal $\mathbf{H}$, sino además del vector recibido $\mathbf{y}$. Para este segundo caso se desarrolla el algoritmo SPA [18, 19], el cual encuentra la ordenación óptima teniendo en cuenta la matriz de canal y el vector recibido. Asociado a este proceso de ordenación y siguiendo una metodología similar a la de cualquier detector SIC, los autores de este algoritmo también encuentran una solución sub-óptima y proponen un mecanismo iterativo para alcanzar la detección óptima ML, que como se verá en capítulos posteriores es claramente competitiva con otros detectores.

Este algoritmo es motivo de estudio de esta tesis doctoral y los resultados de prestaciones y complejidad e implementación VLSI se expondrán detalladamente en capítulos posteriores.

2.3.2.4. Consideraciones sobre la obtención del *soft output*

En la naturaleza de los algoritmos de esta familia, estimación lineal de la salida, está la obtención de un único candidato, *hard output*. Este hecho dificulta enormemente la obtención un LLR capaz de generar una salida *soft output* de calidad. En la literatura especializada aparecen algunas propuestas para generar *soft output* a través de la modificación de estos algoritmos y/o combinación con otros algoritmos [31], o la obtención de *soft output*, pero partiendo de *soft input*[32, 33], sin embargo, no dejan de ser meras aproximaciones para resolver un problema intrínseco al algoritmo.

El algoritmo SPA, sin embargo, sí deja una puerta abierta para la obtención del *soft output* con mayor calidad, ya que sus autores proponen un mecanismo iterativo para alcanzar la solución ML, a través del cual se obtienen candidatos con los que generar una lista para obtener el LLR.

2.3.3. Algoritmos de búsqueda en árbol

Los algoritmos de búsqueda en árbol alcanzan las prestaciones ML o cuasi ML con una complejidad computacional en general mucho más baja que la búsqueda exhaustiva. En general son buenos candidatos para la implementación VLSI y como se verá en esta tesis son claros competidores del algoritmo SPA.

El punto de partida de estos algoritmos está en la distancia euclidea objeto de minimización $d(\mathbf{s_c})$ de la ecuación 2.13, cuyo desarrollo matricial se muestra a continuación:

$$d(\mathbf{s_c}) = \| \begin{bmatrix} \widehat{y}_1 \\ \widehat{y}_2 \\ \vdots \\ \widehat{y}_{M_t} \end{bmatrix} - \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1M_t} \\ 0 & r_{22} & \dots & r_{2M_t} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & r_{M_tM_t} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_{M_t} \end{bmatrix} \|^2, \qquad (2.19)$$

La estructura triangular de este sistema de ecuaciones invita a descomponer el problema de la búsqueda de el mejor vector en un problema recursivo de búsqueda de vectores candidatos parciales $\mathbf{s}_c^{(k)} = [s_k, s_{k+1}, \dots, s_{M_t}]$. Estos vectores, como muestra la figura 2.4, se asocian a los nodos de un árbol donde el paso de un nivel al siguiente supone añadir un nuevo símbolo en el vector, con su contribución sumativa a la distancia parcial $d_k(\mathbf{s}_c^{(k)})$. El objetivo de un detector ML es alcanzar la base del árbol (nivel k=1) consiguiendo el vector cuya distancia es la menor de entre todos.

La figura 2.4 muestra cómo se forma el árbol de un sistema MIMO 3x3 BPSK. Al nodo de nivel $k = 4$ se le asigna el valor de distancia $d_4 = 0$. Para avanzar un nivel (del 4 al 3) en el árbol, se incrementa la distancia con un valor e_3. Y así sucesivamente hasta llegar a la base del árbol, tal y como se

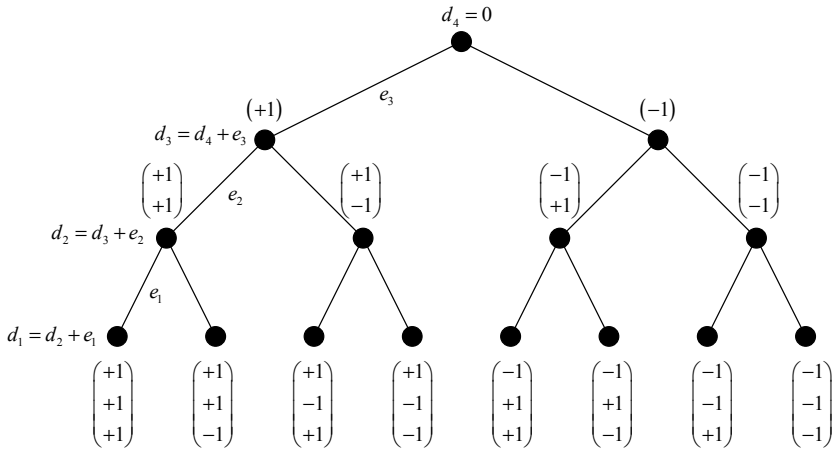


Figura 2.4: Árbol de búsqueda para un sistema MIMO 3x3 BPSK

muestra en las siguientes ecuaciones particularizadas para este ejemplo:

$$\begin{aligned} e_1 = |\widehat{y}_1 - r_{11}s_1 - r_{12}s_2 - r_{13}s_3|^2 \Rightarrow d_1 = d_2 + e_1 \\ e_2 = |\widehat{y}_2 - r_{22}s_2 - r_{23}s_3|^2 \Rightarrow d_2 = d_3 + e_2 \\ e_3 = |\widehat{y}_3 - r_{33}s_3|^2 \Rightarrow d_3 = d_4 + e_3 \\ d_4 = 0 \end{aligned}$$

Para un sistema MIMO $N_r \times M_t$ q-QAM las ecuaciones de recursividad de un algoritmo de búsqueda en árbol son las siguientes:

$$d_k(\mathbf{s}_c^{(k)}) = d_{k+1}(\mathbf{s}_c^{(k+1)}) + e_k(\mathbf{s}_c^{(k)}), \tag{2.20}$$

donde los incrementos positivos de la distancia $|e_k(\mathbf{s}_c^{(k)})|^2$ vienen dados por:

$$\begin{aligned} e_k(\mathbf{s}_c^{(k)}) &= |b_{k+1}(\mathbf{s}_c^{(k+1)}) - R_{kk}s_k|^2 \\ b_{k+1}(\mathbf{s}_c^{(k+1)}) &= \widehat{y}_k - \sum_{j=k+1}^{M_t} R_{kj}s_j. \end{aligned} \tag{2.21}$$

Cuando $N_r = M_t$, la recursión empieza desde $d_{M_t+1}(\mathbf{s}_c^{(M_t+1)}) = 0$ en el origen del árbol. Sin embargo, si $N_r > M_t$:

$$d_{M_t+1}(\mathbf{s}_c^{(M_t+1)}) = cte \quad \text{con} \quad cte = \|\mathbf{Q}_2^T \mathbf{y}_c\|^2, \tag{2.22}$$

según se ha visto en la ecuación 2.12 de la sección 2.2.2.

Estas ecuaciones son la base para el desarrollo de diversas estrategias de búsqueda en árbol, en particular hay dos que destacan especialmente sobre

el resto de algoritmos: Shere Decoding (SD) y KBest. Ambas tienen sus orígenes en [5] y aplicación práctica para comunicaciones inalámbricas en [3] y [6].

Dependiendo de la estrategia de búsqueda utilizada, los algoritmos correspondientes alcanzan o no alcanzan la solución ML. El detector SD, por ejemplo, consigue alcanzar la detección ML si el tiempo de búsqueda no está restringido. Sin embargo, el detector KBest, con un coste computacional moderado, no alcanza la detección ML.

2.3.3.1. Sphere Decoding

El algoritmo Sphere decoding (SD), matemáticamente fundamentado en [5] y ampliamente mejorado desde su primera aplicación en [3] por [8, 34, 12], adopta como idea fundamental la reducción del número de candidatos a ser considerados para la búsqueda de la solución ML. Con este propósito, la búsqueda se reduce a aquellos vectores cadidatos $\mathbf{s_c}$ para los cuales $\mathbf{H_c s_c}$ está dentro de una hiperesfera centrada en $\mathbf{y_c}$ y con radio *rad* como ilustra la figura 2.5. La inecuación correspondiente es:

$$\|\mathbf{y_c} - \mathbf{H_c s_c}\|^2 < rad^2, \tag{2.23}$$

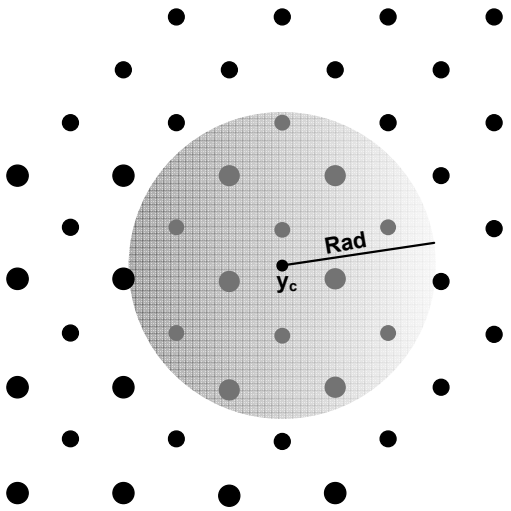


Figura 2.5: *Lattice* con esfera centrada en $\mathbf{y_c}$ y radio Rad.

Por lo tanto, el reto de alcanzar la solución ML de la ecuación 2.13 ahora pasa por evaluar aquellos vectores candidatos cuya distancia esté dentro de

los límites de la esfera. La reducción de complejidad es clara si tenemos en cuenta que el cálculo de la distancia se realiza de forma recursiva sumando distancias parciales más pequeñas segun 2.20. Sin pérdida de generalidad, por lo tanto, se puede decir que si se cumple la inecuación 2.23 también se cumplirá:

$$d_k(\mathbf{s}_c^{(k)}) < rad^2. \tag{2.24}$$

De esta manera, si se alcanza un nodo del árbol cuya distancia parcial viole los límites de la esfera, automáticamente todos los nodos y ramas hijos de éste violarán también los límites de la esfera y por lo tanto, no será necesario evaluar sus distancias parciales.

Sobre el algoritmo SD cabe hacer las siguientes consideraciones:

Anchura/Profundidad: En la búsqueda de candidatos que satisfagan los límites de la esfera, el árbol puede ser recorrido en sentido horizontal (a lo ancho y siempre avanzando hacia el nivel mas bajo $k = 1$) o en sentido vertical (en profundidad y subiendo y bajando). En general para la implementación del SD es más eficiente atravesar el árbol en profundidad debido a los bajos requerimientos de memoria que necesita éste para su implementación y a la gran sensibilidad que presenta el sentido horizontal a la elección del radio.

Reducción de Radio: La técnica de reducción del radio es extremadamente eficiente para acelerar la poda del árbol en la búsqueda en profundidad sin comprometer para nada las prestaciones de BER. La idea básica es empezar la búsqueda con un radio inicial $rad = rad_0$ y ajustar el radio de la esfera a

$$rad^2 \leftarrow d(\mathbf{s}_c) \tag{2.25}$$

cuando el detector encuentra una solución. El hecho de reducir la esfera según 2.25 supone eliminar las ramas del árbol que exceden los límites de la esfera. La solución ML se alcanza cuando la esfera sólo contiene una única solución. Una ventaja adicional de la reducción dinámica del radio es el alivio que se produce en la elección del radio inicial, ya que como se muestra en [34], uno puede simplemente elegir $rad_0 = \infty$ sin penalización en la complejidad.

Ordenación Schorr/Euchner (SE): Sin reducción de radio, el orden con que los hijos de un nodo son explorados no tiene influencia en la complejidad, ya que la elección del radio inicial determina los nodos que serán podados del árbol. Sin embargo, SD con reducción de radio puede ser más eficiente si se suplementa con una estrategia de selección de hijos con la mejor métrica [9, 8]. La idea básica detrás de la ordenación Schnorr/Euchner (SE) es dar preferencia a los nodos con

menor distancia parcial para así alcanzar antes la solución que reduce el radio. El precio que hay que pagar por alcanzar antes la solución es una penalización en la complejidad, ya que se requieren mecanismos de ordenación.

2.3.3.2. KBest

Merecen especial atención los algoritmos de búsqueda en anchura basados en el clásico "M algorithm"[7]. Las aplicaciones de este algoritmo para la detección de sistemas MIMO han tenido diferentes denominaciones en la literatura especializada, siendo las más destacadas "KBest"[6], utilizada en este trabajo, y "QR Descomposition and M-algorithm" (QRD-M) [35].

Como ya se ha comentado en el apartado anterior, los algoritmos de búsqueda en anchura recorren el arbol nivel por nivel avanzando en una sola dirección y sin vuelta atrás, rechazando todos los nodos excepto K, aquellos con la mejor métrica, antes de avanzar al siguiente nivel. La poda del árbol se realiza limitando el número de nodos admisibles, en cada nivel del árbol, al parámetro de diseño K. La expansión del árbol se realiza extendiendo cada uno de estos nodos con q nuevas ramas, obteniéndose en el siguiente nivel qK nuevos nodos de los cuales de nuevo se seleccionarán los K con mejor métrica (distancia parcial). De esta manera, solamente son visitados en cada nivel qK nodos, y sólamente es necesario almacenar K distancias parciales entre un nivel y el siguiente. La selección del conjunto de nodos admisibles se realiza siguiendo el criterio de la menor distancia parcial asociada. La figura 2.6 muestra este proceso de forma esquemática para un sistema MIMO complejo. El mecanismo es el mismo para un sistema MIMO real con $M = 2M_t$ niveles y expansión de $b = log_2(q)$ nuevos nodos por cada K nodos seleccionados en un nivel.

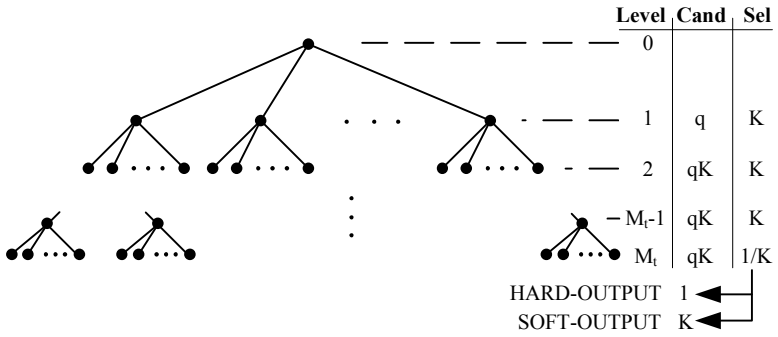


Figura 2.6: Algoritmo de detección KBest

El objetivo del algoritmo es alcanzar la solución *hard output* seleccionan-

do en el último nivel el candidato de métrica menor, u obtener la solución *soft output* calculando el LLR a partir de la lista de los mejores K candidatos obtenidos en el último nivel, tal y como se muestra en la figura 2.6.

Cuando se utilizan ambos mecanismos, KBest y radio de búsqueda, K sólo define el límite superior de los nodos admisibles en un nivel. En la mayoría de las implementaciones prácticas sólo se usa la restricción de la K. De hecho, estrictamente hablando, los decodificadores KBest no son Sphere Decoders.

2.3.3.3. Consideraciones sobre la obtención del *soft output*

Los dos detectores, SD y KBest, pueden generar una lista de candidatos útiles para la obtención de la salida *soft output*, sin embargo, hay una gran diferencia entre los candidatos generados por ambos. Mientras que SD genera una lista con los mejores, aquellos que más se aproximan a la solución ML, los candidatos que genera el algoritmo KBest no sólo no son los mejores, en el sentido anteriormente descrito, sino que están altamente correlados, es decir, gran parte de los vectores candidatos comparten muchos símbolos. Este hecho repercute negativamente en la calidad de la detección *soft output* de este algoritmo.

2.3.4. Comparativa de Prestaciones vs Complejidad

La figura 2.7 muestra un esquema en forma de árbol donde se clasifican los diferentes detectores MIMO en función de su capacidad para alcanzar o no lo solución ML. La parte derecha del árbol la ocupan los detectores que alcanzan la solución ML, bien a través de búsqueda exhaustiva (sec 2.3.1), o bien a través de algoritmos de búsqueda en árbol (Sphere Decoding, sec 2.3.3.1). En la parte contraria (izquierda del árbol), se encuentran los algoritmos de detección lineal con prestaciones alejadas de la detección ML pero, como contrapartida, con muy baja complejidad como el lector podrá observar a lo largo de esta sección. La parte central del árbol la ocupan los algoritmos SIC (sec 2.3.2) y KBest 2.3.3.2), ambos con vocación de alcanzar la solución ML. En esta sección se exponen con detalle las características de prestaciones BER y complejidad computacional de los siguientes detectores: ZF, ZF-MMSE, VBLAST, KBest y ML (búsqueda exhaustiva). El estudio del algoritmo SPA así como su comparativa con el KBest se realiza con profundidad en los capítulos posteriores.

- **Busqueda exhaustiva:** Es un buen ejercicio para el lector, y un baño de realidad, comenzar este estudio con una comparativa de la complejidad computacional de una búsqueda exhaustiva. La figura 2.8 muestra la evolución del número de candidatos a evaluar para

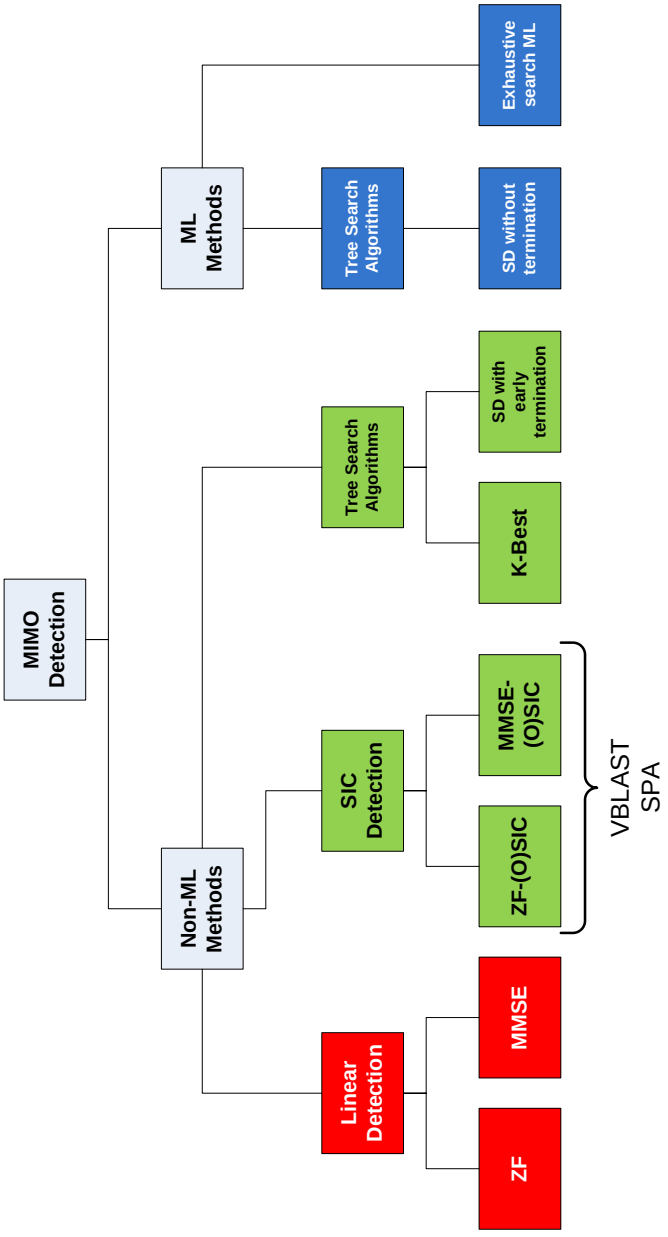


Figura 2.7: Comparativa de detectores MIMO

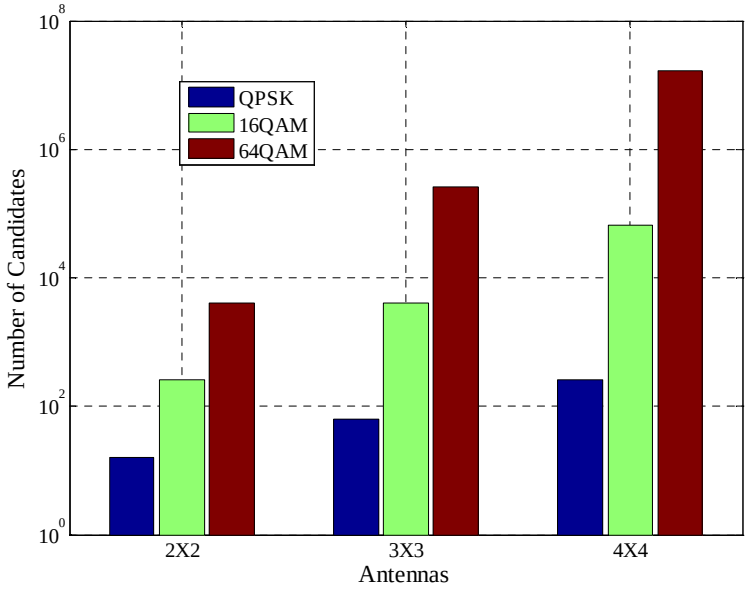


Figura 2.8: Comparativa de complejidad del detector ML de búsqueda exhaustiva en función de antenas y modulaciones

diferentes sistemas MIMO en función del número de antenas y de la modulación. Para un sistema MIMO $M_t \times M_t$, $q-QAM$, el número de candidatos a evaluar es q^{M_t}.

Para cada candidato hay que evaluar la expresión 2.3 ($d(\mathbf{s_c}) = \|\mathbf{y_c} - \mathbf{H_c s_c}\|^2$), y retener el candidato cuya métrica sea menor.
Se observa una enorme variabilidad de las dimensiones del problema, sobre todo en función de la modulación: lo que podría ser factible para QPSK, es inviable para 64-QAM y ni que decir, aunque no esté representado en esta gráfica, para 256-QAM, modulación estudiada en capítulos posteriores de esta tesis.

- **Detectores lineales y VBLAST:** Los métodos basados en la detección lineal, como ya se ha explicado en la sección 2.3.2, consideran la relación de entrada-salida de un sistema MIMO (ecuación 2.1) como una estimación lineal sin restricciones del problema, cuya solución es alcanzada a través de mínimos cuadrados (ZF) o de la minimización del error cuadrático medio (MMSE). La figura 2.9 muestra las prestaciones BER alcanzadas por los detectores ZF y MMSE para un sistema MIMO 4x4 16-QAM. Las diferencias son mínimas y sólo se aprecian diferencias cuando la SNR aumenta sustancialmente.

  La figura 2.10 muestra una comparativa entre las prestaciones BER

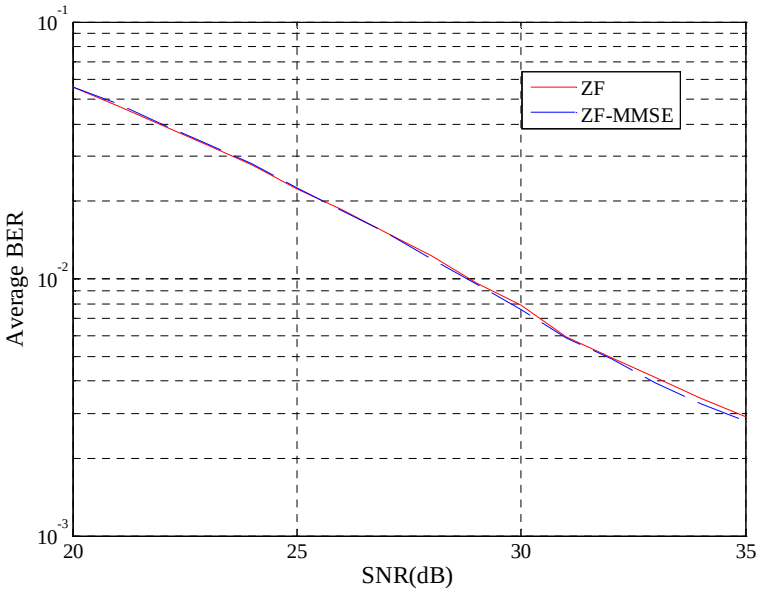


Figura 2.9: Comparativa prestaciones BER de detectores MIMO 4x4 16-QAM. Detección ZF y ZF-MMSE

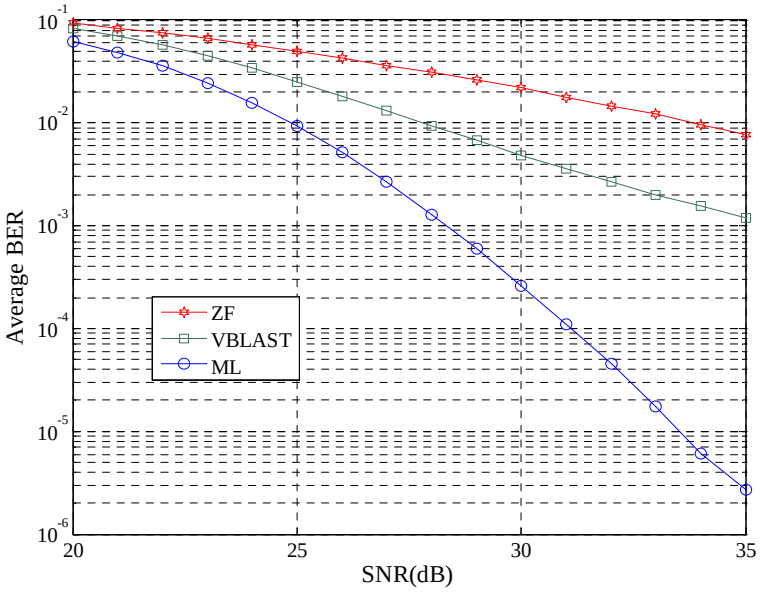


Figura 2.10: Comparativa prestaciones BER de detectores MIMO 4x4 64-QAM. Detección lineal y VBLAST

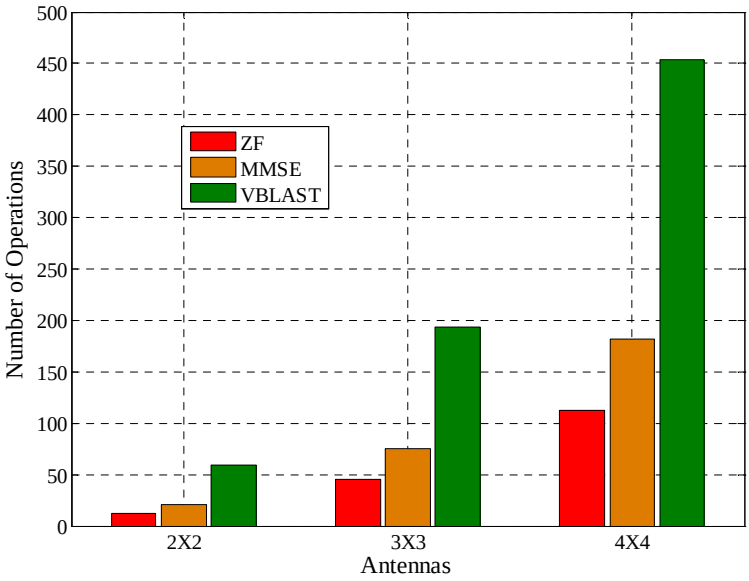


Figura 2.11: Comparativa de complejidad de detectores lineales y SIC en función del número de antenas para un sistema MIMO q-QAM

alcanzadas por un detector ZF y un detector SIC con ordenación VBLAST para un sistema MIMO 4x4 64-QAM. Las prestaciones BER del detector ML muestran el objetivo a alcanzar por cualquier detector. Es destacable la mejora del detector VBLAST sobre el ZF en prestaciones BER, lo que se traduce también en un aumento de la complejidad.

La figura 2.11 muestra una comparativa de la complejidad de los detectores ZF, MMSE y VBLAST en función del número de antenas de un sistema MIMO q-QAM. Las metricas de complejidad, obtenidas de la tesis doctoral de A. Burg [11], contemplan en todos los casos las operaciones de preprocesado de descomposición QR de la matriz H necesarias para la inversión de la matriz de canal. Se consideran operaciones el número de multiplicaciones y el número de rotaciones vectoriales. El esquema de detección V-BLAST [2] realiza una detección SIC basándose en una ordenación de la matriz de canal que optimiza la BER. El algoritmo original tiene una complejidad computacional que crece con un orden de $M_t^3 N_r$, sin embargo, Hassibi en [36] propone una implementación más eficiente con orden $M_t^2 N_r$, el cual se utiliza en las comparativas de este trabajo.

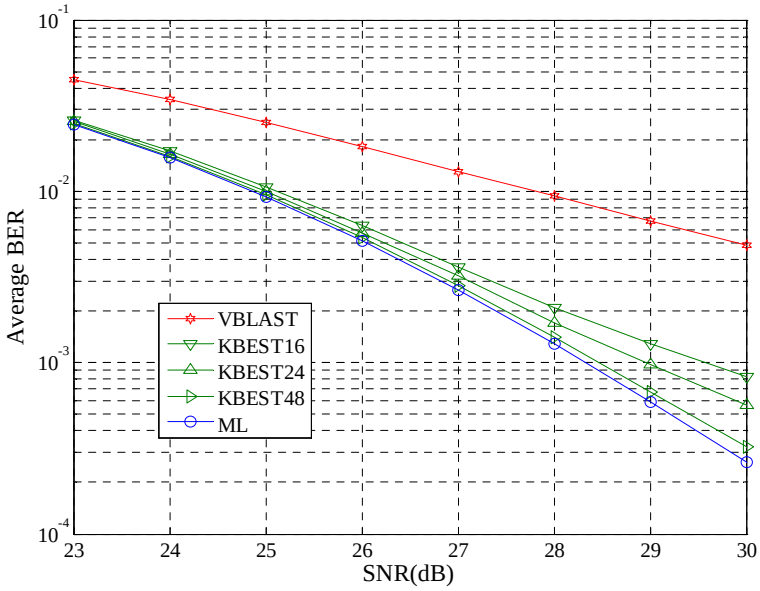


Figura 2.12: Evolución de prestaciones BER del detector KBest en función de K. MIMO 4x4 64-QAM.

Los algoritmos de detección lineal y SIC tienen una complejidad claramente asumible para una implementación hardware, sin embargo, las prestaciones BER alcanzadas son mediocres y alejadas de las ML. Es objetivo de esta tesis es mejorar estas prestaciones a través de un algoritmo SIC.

- **Detector KBest:** Como ya se ha comentado en la sección 2.3.3.2 el detector KBest, aunque está basado en los algoritmos de búsqueda en árbol, en su concepción original tiene una complejidad computacional fija en función del parámetro K, lo que lo hace apto para su implementación y comparación con otros algoritmos también de complejidad computacional fija. La figura 2.12 muestra la evolución de las prestaciones BER de un detector KBest para un sistema MIMO 4x4 64-QAM en función del parámetro K. Se observa cómo a medida que aumenta K (16,24,48) el detector se acerca al ML.

  La evolución de la complejidad computacional con la K puede obsevarse en la figura 2.13, donde se han considerado como operaciones las multiplicaciones y comparaciones, lo cual es crítico en estos detectores debido a la necesidad de ordenar grandes listas de datos a medida que crece el parámetro K.
  La figura 2.14 muestra una comparativa de complejidad computacio-

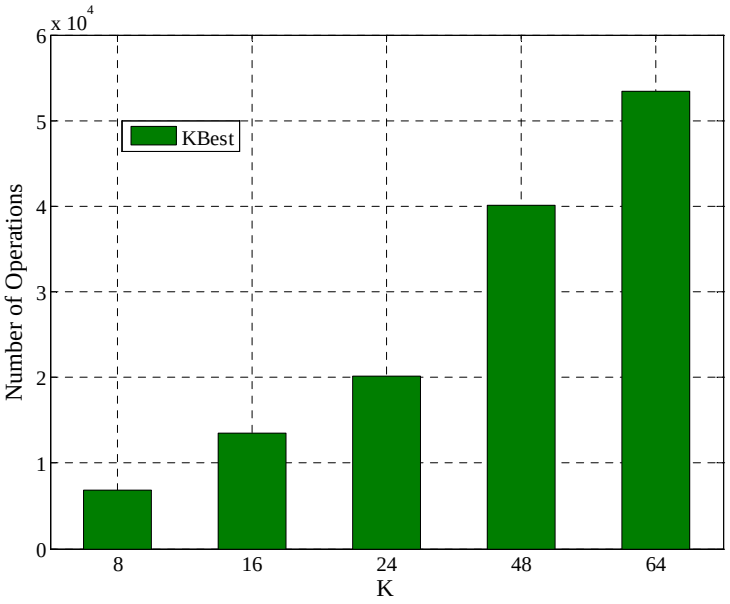


Figura 2.13: Comparativa de complejidad de detectores KBest para un sistema MIMO 4x4 64QAM en función del parámetro K

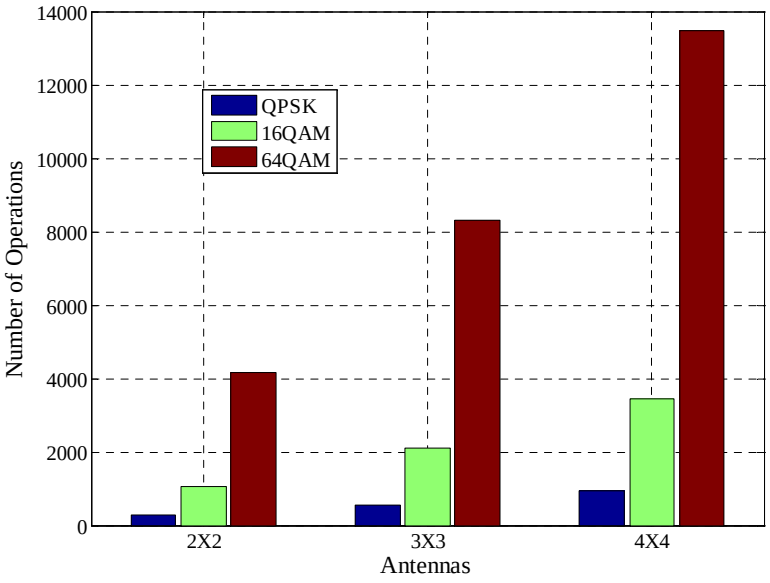


Figura 2.14: Comparativa de complejidad del detector KBest (K=16) en función de antenas y modulaciones

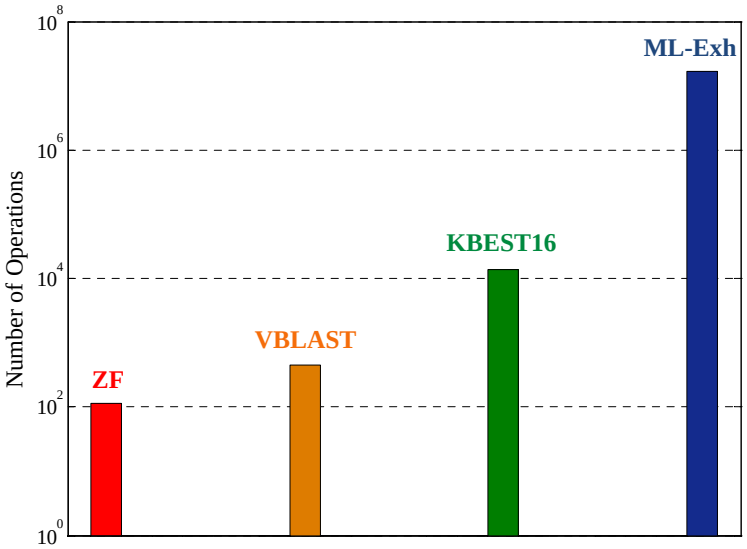


Figura 2.15: Comparativa de complejidad de detectores para un sistema MIMO 4x4 64QAM

nal de detectores KBest en función del número de antenas y del esquema de modulación del sistema MIMO, manteniendose fijo el parámetro K (K=16). Se observa que para una configuración de antenas determinada y una K fija, la complejidad de los detectores KBest crece exponencialmente con el esquema de modulación utilizado.

Finalmente, la figura 2.15 muestra una comparativa de la complejidad computacional de todos los detectores tratados en esta sección. El lector puede obsevar la diferencia del orden de magnitud de complejidad computacional entre los diferentes detectores que se corresponde con las prestaciones BER alcanzadas por cada uno de ellos.

En capítulos posteriores se desarrolla un detector SIC iterativo que compite en prestaciones BER y complejidad con los detectores KBest presentados en este capítulo.

2.4. Conclusiones

El incremento de la eficiencia espectral provocado por la diversidad espacial de los sistemas de comunicaciones basados en tecnología MIMO, se modela matemáticamente mediante un sistema matricial, cuyas dimensiones son directamente proporcionales al número de antenas transmisoras y

al número de antenas receptoras.

La detección de las señales MIMO se realiza en dos fases: preprocesado y detección de símbolos. En la primera se realizan todas las operaciones necesarias, fundamentalmente sobre la matriz de canal, sólamente cuando hay un cambio en las condiciones del canal. La segunda comprende todas aquellas operaciones necesarias para realizar la estimación de los símbolos transmitidos, a partir del vector recibido.

Existen dos modelos para la detección de símbolos transmitidos: la detección *hard output* y la detección *soft output*. El primer modelo encuentra la estimación del vector de símbolos transmitidos, siendo la mejor estimación, la solución óptima (ML), es decir, aquella que ha sido transmitida con mayor probabilidad. La detección *soft output*, sin embargo, encuentra la probabilidad de acierto o fallo (APP) en la detección de cada bit, y esta información es utilizada por los sistemas de corrección de errores (FEC).

Para obtener las salidas *hard output* y/o *soft output* existen tres grandes familias de detectores MIMO basados en diferentes algoritmos: los detectores basados en búsqueda exhaustiva, los detectores lineales y de cancelación sucesiva de interferencias (SIC) y los detectores no lineales o de búsqueda en árbol:

- Los primeros comprueban uno a uno todos los candidatos posibles, y por lo tanto, alcanzan la solución óptima ML, sin embargo, el coste computacional para su implementación VLSI sólo es asumible para sistemas MIMO de baja diversidad, como máximo 2x2 16-QAM. Cuando crece el número de antenas en transmisión y recepción y/o el esquema de modulación empleado, la implementación de este detector es inviable. Un aspecto positivo de este detector es que puede generar una lista con los mejores candidatos, los que se acercan más a la solución óptima, que puede ser utilizada para obtener la salida *soft output*.

- Los detectores lineales y SIC tienen una complejidad independiente del esquema de modulación utilizado y claramente asumible para una implementación *hardware*, sin embargo, las prestaciones BER alcanzadas son mediocres y alejadas de las ML. Subyacente a la naturaleza de estos algoritmos, estimación lineal de la salida, está la obtención de un único candidato, *hard output*. Este hecho dificulta enormemente la obtención un LLR capaz de generar una salida *soft output* de calidad.

- Los detectores no lineales o de búsqueda en árbol se subdividen en dos categorías: *sphere decoding* (SD) y KBest. El primero encuentra la solución ML, con un coste computacional muy razonable, pero presenta como principal inconveniente para su implementación *hardware* que la tasa de detección es variable. El algoritmo KBest resuelve este problema a costa de alcanzar una solución sub-óptima, y puede acercarse a

Tabla 2.1: Resumen cualitativo de prestaciones de detectores MIMO

	B. Exh.	Lineal	SIC	SD	KBest
Complejidad	ALTA	MUY BAJA	BAJA	BAJA	MEDIA
Prestaciones BER	ML	MUY BAJA	BAJA	ML	MEDIA
Tasa de Detección	FIJA	FIJA	FIJA	VARIABLE	FIJA
Calidad *Soft Output*	ALTA	BAJA	BAJA	ALTA	MEDIA

la solución óptima ML incrementando su complejidad computacional a través del parámetro K. El principal inconveniente de estos detectores, a diferencia de los SIC, es el crecimiento que experimenta el árbol, sobre el que se realiza la búsqueda, con la diversidad del sistema MIMO utilizado, especialmente con el esquema de modulación. Este hecho se traduce en un incremento del tiempo de búsqueda (tasa de detección) en los detectores SD y en un crecimiento exponencial de la complejidad computacional en los detectores KBest.

Ambos detectores, SD y KBest, pueden generar una lista de candidatos útiles para la obtención de la salida *soft output*, sin embargo, hay una gran diferencia entre los candidatos generados por ambos. Mientras que SD genera una lista con los mejores, aquellos que más se aproximan a la solución ML, los candidatos que genera el algoritmo KBest no sólo no son los mejores, en el sentido anteriormente descrito, sino que están altamente correlados, es decir, gran parte de los vectores candidatos comparten muchos símbolos. Este hecho repercute negativamente en la calidad de la detección *soft output* de este algoritmo.

En la tabla 2.1 se resumen las prestaciones alcanzadas por los diferentes decodificadores estudiados en este capítulo, desde un punto de vista cualitativo. Para establecer comparación entre ellos se han seleccionado cuatro parámetros de diseño determinantes para la implementación *hardware*. El primero de ellos clasifica los detectores teniendo en cuenta la suma contributiva de su complejidad computacional y de la evolución de la misma con la diversidad del sistema MIMO, es decir, del número de antenas y esquema de modulación seleccionados. El segundo, en cuanto a las prestaciones BER alcanzadas, siendo ML la detección óptima. El tercer parámetro de diseño es la variabilidad de la tasa de detección en función de la SNR o diversidad del sistema, y en último lugar se realiza la comparación en función de la calidad del LLR, obtenido de cada decodificador, para la detección *soft output*.

Tomando como punto de partida de este trabajo el estudio exhaustivo que se ha realizado en este capítulo de los diferentes detectores MIMO, y a la vista de los trabajos publicados sobre el nuevo algoritmo de proyecciones sucesivas SPA [18, 19], procede ahora realizar una valoración previa sobre

las características potenciales del mismo y su idoneidad para una posible implementación *hardware*.

El algoritmo SPA pertenece a la familia de los detectores de cancelación sucesiva de interferencias (SIC) lo cual es sinónimo de baja dependencia de la complejidad computacional con la diversidad del sistema MIMO y tasa de detección independiente de la SNR. El hecho de que los autores propongan un mecanismo para alcanzar de forma iterativa las prestaciones BER óptimas, ya de por sí constituye un estímulo para su evaluación, ya que uno de los principales problemas de los algoritmos SIC es la mediocridad de la BER conseguida. Además, el hecho de que a través de este mecanismo iterativo se consigan diferentes candidatos podría solucionar en todo o en parte la carencia que presentan los algoritmos SIC en cuanto a *soft output* se refiere.

A la vista de estos resultados, se considera procedente evaluar las prestaciones del algoritmo SPA como fase previa a la búsqueda del equilibrio, entre diferentes parámetros de diseño, necesario para una potencial implementación *hardware*.

Capítulo 3

Algoritmo de Proyección Sucesiva (SPA)

Incluso utilizando la enumeración de candidatos más efectiva (Schnorr-Euchner), el coste computacional de los algoritmos de búsqueda en árbol, es altamente sensible a la ordenación de las columnas de la matriz de canal [34]. Este comportamiento se debe a la naturaleza generalmente no unitaria del *lattice* asociado al problema de detección MIMO.

Resulta natural considerar la propagación de errores desde la perspectiva de los algoritmos SIC, en los cuales no existe la oportunidad de reconsiderar las decisiones tomadas. De esta manera los errores se propagan desde un estado de detección al siguiente empobreciéndose de forma considerable las prestaciones de los detectores. Sin embargo, en el contexto de las estrategias de búsqueda en árbol, como Sphere Decoder, donde el detector siempre alcanza la solución ML, los errores no se propagan como tal. En su lugar, una decisión equivocada provoca la búsqueda en aquellas ramas donde no está la solución ML, incrementando el coste computacional de la búsqueda.

En [18] se realiza un estudio del impacto que tiene la ordenación sobre la eficiencia computacional de los algoritmos de búsqueda en árbol. A diferencia de otros trabajos realizados previamente donde el criterio de ordenación se basa exclusivamente en la matriz de canal [34, 2], en [18] se demuestra que la ordenación óptima para un detector ML depende no sólo de la matriz de canal sino, además, del vector recibido, objeto de decodificación. Desde una perspectiva geométrica los autores de este trabajo desarrollan un algoritmo eficiente para computar la ordenación óptima de las columnas de la matriz

de canal, aplicable en la fase de pre-procesado de cualquier detector.

Cuando el algoritmo de ordenación descrito en [18] termina, no sólo se consigue la ordenación óptima sino que además, en paralelo, se alcanza una solución inducida por esta ordenación. El mecanismo para obtener esta solución sub-óptima, se denomina "*Successive Projection Algorithm*" (SPA) y se describe en [19]. Es objeto de esta tesis el estudio de viabilidad y la adaptación de este algoritmo para la implementación VLSI.

En la sección 3.1 se exponen los fundamentos básicos para el desarrollo del algoritmo SPA desde una perspectiva geométrica. En la sección 3.2 se explica el funcionamiento del algoritmo, y aunque es un detector sub-óptimo, en la sección 3.3 se muestra cómo se puede alcanzar la solución ML a través de la aplicación sucesiva del algoritmo. Además en esta sección se presentan los mecanismos que hacen posible la implementación hardware de este detector. La sección 3.4 describe los algoritmos necesarios para obtener las salidas hard output y soft output de los detectores SPA y finalmente, en la sección 3.5 se comparan las prestaciones y complejidad computacional alcanzadas por este detector con otros publicados en la literatura especializada, y cuyos resultados de implementación son altamente competitivos.

3.1. Geometría MIMO. Fundamentos

En el capítulo anterior se ha considerado la detección MIMO ML desde una perspectiva algebraica. Tomando como punto de partida la descomposición QR de la matriz de canal, el problema se ha triangularizado y la función objeto de optimización (2.13) se ha descompuesto en suma de distancias parciales (2.20). Desde este punto de vista el sistema se ha dibujado como una estructura en árbol donde el movimiento entre los distintos nodos se realiza sumando o restando distancias parciales. El problema de la detección MIMO también puede ser interpretado desde una perspectiva geométrica con una estructura en árbol asociada, equivalente a la descrita en la sección 2.3.3, que da lugar, análogamente, a una decodificación eficiente. De hecho, mediante esta perspectiva se añaden herramientas geométricas que complementan la descripción algebraica mediante la visualización del problema. Para conseguir la misma estructura sin utilizar la descomposición QR es necesario introducir una serie de conceptos y definiciones geométricas aplicadas sobre el sistema MIMO real descrito por la ecuación 2.7, que se exponen a continuación.

Sub-espacio: Dada la matriz $\mathbf{H}$ de rango M, el sub-espacio generado por las columnas de $\mathbf{H}$ se define como:

$$S(\mathbf{H}) \triangleq \{\mathbf{z} \mid \mathbf{z} = \mathbf{H}\mathbf{a}, \mathbf{a} \in \mathbb{R}^M\}. \tag{3.1}$$

Encontrar la solución ML implica trabajar dentro de este espacio en la búsqueda del punto más cercano al objetivo $\mathbf{y}$.

***Lattice*:** Dada la matriz $\mathbf{H}$ de rango M y el alfabeto $\mathcal{X}$ de tamaño $\mathcal{B}$, se define la red de puntos (*lattice*) dentro del conjunto búsqueda como:

$$\mathcal{L}(\mathbf{H}, \mathcal{X}) \triangleq \{\lambda \mid \lambda = \mathbf{Hs}, \mathbf{s} \in \mathcal{X}^M\}, \tag{3.2}$$

donde $\mathbf{H}$ es la matriz generadora del *lattice* y sus columnas $\mathbf{h}_j$ se denominan vectores base. En función de si $\mathcal{X}$ es un conjunto finito o infinito, entonces el *lattice* también será finito ó infinito.
Hay $\mathcal{B}^M$ puntos en el *lattice* $\mathcal{L}(\mathbf{H}, \mathcal{X})$, 4 puntos, como muestra la figura 3.1.I para el caso de sistema MIMO 2x2 BPSK ($\mathcal{B} = 2$, $\mathcal{X} = \{+1, -1\}$, $M = 2$).

Conjunto afín: Asociado a cada columna i de la matriz generadora de *lattice* existen $\mathcal{B}$ conjuntos afines definidos por:

$$\mathcal{F}_i(s_i) \triangleq \{\mathbf{z} | \langle \mathbf{z} - \mathbf{h}_i s_i, \mathbf{g}_i \rangle = 0\}, \tag{3.3}$$

donde $s_i \in \mathcal{X}$, $\mathbf{g_i}$ son las filas de la matriz $\mathbf{G}$ (estimación ZF de $\mathbf{H}, \mathbf{G} = \mathbf{H}^{-1}$) y $\langle \cdot, \cdot \rangle$ representa el producto escalar.
Sobre estos conjuntos afines podemos hacer las siguientes observaciones:

1. Son subespacios $M - 1$-dimensionales. El ejemplo de la figura 3.1.I representa un *lattice* bidimensional (un plano formado por las dos columnas de $\mathbf{H}$). Por lo tanto los conjuntos afines son rectas.
2. Asociado a cada columna de la matriz generadora del *lattice* existe una colección de $\mathcal{B}$ conjuntos afines paralelos. En el ejemplo de la figura dos rectas paralelas por cada columna de la matriz $\mathbf{H}$.
3. Desde un punto de vista algebraico $\mathcal{F}_i(s_i)$ contiene puntos posibles del *lattice* que satisfacen la restricción que impone el alfabeto $\mathcal{X}$. Cada conjunto afín contiene $\mathcal{B}^{M-1}$ puntos del *lattice*, y es en las intersecciones de estos sub-espacios donde se encuentran las posibles soluciones de problema MIMO.

A la vista de la definición de los conjuntos afines, parece lógico pensar que un parámetro determinante a la hora de realizar la detección dentro de este espacio geométrico será la distancia desde el objetivo $\mathbf{y}$ hasta los diferentes conjuntos afines. Desde un punto de vista geométrico tiene sentido hablar de la distancia ortogonal. Para ello se realizan las siguientes definiciones:

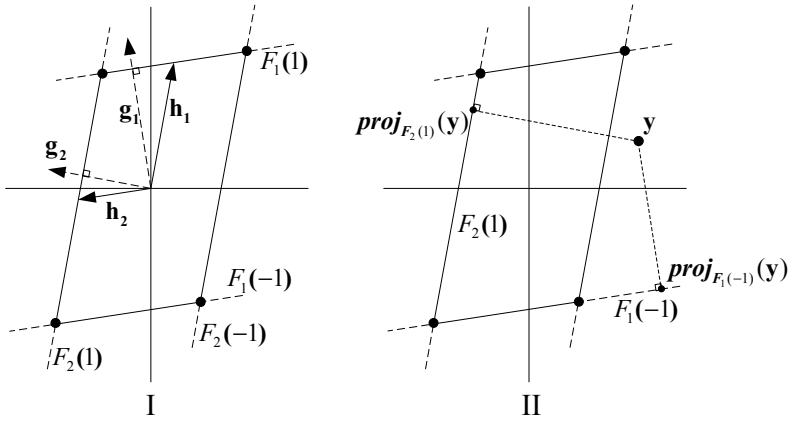


Figura 3.1: Interpretación geométrica del algoritmo SPA, para un sistema MIMO 2x2 BPSK.

Proyección: La proyección ortogonal de un vector $\mathbf{y}$ sobre el conjunto afín $\mathcal{F}_i(s_i)$ se define como:

$$proj_{\mathcal{F}_i(s_i)}(\mathbf{y}) \triangleq \mathbf{y} - \frac{\langle \mathbf{y} - \mathbf{h}_i s_i, \mathbf{g}_i \rangle}{|\mathbf{g}_i|^2} \mathbf{g}_i. \tag{3.4}$$

En la figura 3.1.II se muestran las proyecciones del vector $\mathbf{y}$ sobre los conjuntos afines $\mathcal{F}_1(-1)$ y $\mathcal{F}_2(1)$.

Distancia: El cuadrado de la distancia ortogonal que separa el vector $\mathbf{y}$ del conjunto afín $\mathcal{F}_i(s_i)$ es:

$$d\left(\mathbf{y}, \mathcal{F}_i(s_i)\right) \triangleq d\left(\mathbf{y}, proj_{\mathcal{F}_i(s_i)}(\mathbf{y})\right) = \|\mathbf{y} - proj_{\mathcal{F}_i(s_i)}(\mathbf{y})\|^2. \tag{3.5}$$

por lo tanto, la distancia entre un vector y un conjunto afín es la distancia entre el vector y la proyección ortogonal del mismo sobre el conjunto afín.

El camino natural para encontrar la solución ML de un sistema MIMO, tal y como se ha visto en el capítulo anterior, pasa por la descomposición QR de la matriz de canal. De esta manera se consigue una estructura en árbol que facilita el cálculo de las distancias parciales cuya suma total es el objetivo de la minimización.

En la tabla 3.1 se muestra para un sistema MIMO 2x2 las dos alternativas de decodificación: la algebreica, utilizando la descomposición QR (en la parte derecha) y la geométrica (en la parte izquierda).

El lector puede observar cómo ambas alternativas comparten el mismo objetivo (minimizar $d(\mathbf{s})$) y dan lugar a la misma estructura en árbol, al

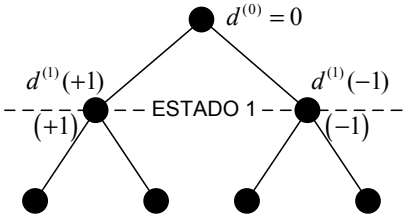
Geométrica $\mathbf{H}$	Algebraica $\mathbf{H} = \mathbf{QR}$
$\mathbf{y} = \mathbf{Hs} + \mathbf{n}$	$\widehat{\mathbf{y}} = \mathbf{Rs} + \mathbf{Q^T n}\ ;\ \widehat{\mathbf{y}} = \mathbf{Q^T y}$
$\hat{\mathbf{s}} = \arg\min_{\mathbf{s}\in(\mathcal{X})^M} d(\mathbf{s})$	
$d(\mathbf{s}) = \|\mathbf{y} - \mathbf{Hs}\|^2$	$d(\mathbf{s}) = \|\widehat{\mathbf{y}} - \mathbf{Rs}\|^2$
$\| \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} - \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \|^2$	$\| \begin{bmatrix} \widehat{y}_1 \\ \widehat{y}_2 \end{bmatrix} - \begin{bmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \|^2$
	
$d^{(1)} = \|\mathbf{y} - proj_{\mathcal{F}_2(s_2)}(\mathbf{y})\|^2$	$d^{(1)} = \|\widehat{y}_2 - r_{22}s_2\|^2$
$\mathbf{y}^{(1)} = proj_{\mathcal{F}_2(s_2)}(\mathbf{y}) - \mathbf{h_2}s_2$	$\widehat{\mathbf{y}}^{(1)} = \widehat{\mathbf{y}} - \mathbf{r_2}s_2$

Tabla 3.1: Alternativas de decodificación de un sistema MIMO 2x2 BPSK

menos aparentemente. Una diferencia sustancial es que en la alternativa algebraica la naturaleza triangular de la matriz fuerza a comenzar la decodificación por el símbolo s_2, es decir, el árbol se construye desde M hasta 1. En la alternativa geométrica no existe esta restricción: el árbol puede construirse empezando desde cualquier nivel y siguiendo cualquier orden. Esto resulta ventajoso ya que las estrategias de decodificación en árbol son especialmente sensibles a la ordenación de la matriz de canal. Tanto es así, que en muchas ocasiones los decodificadores basados en algoritmos de búsqueda en árbol incorporan, en su fase de preprocesado, mecanismos de ordenación de la matriz de canal, bien para acelerar la convergencia en la solución ML, en el caso de *Sphere Decoding*, o bien para mejorar las prestaciones BER en los decodificadores basados en el algoritmo KBest.

Para establecer equivalencias entre la dos alternativas se ha decodificado el símbolo s_2 en ambas (estado 1). Como consecuencia, para el cálculo de

las distancias en la alternativa geométrica se realizan las proyecciones de $\mathbf{y}$ sobre los conjuntos afines a la columna 2 ($\mathbf{h_2}$) de la matriz de canal.

La idea de reducción del *lattice* no se suele utilizar en la literatura especializada cuando se describen detectores en árbol. Sin embargo, resulta evidente que una vez detectado el símbolo s_2, el sistema evoluciona mediante una reducción natural del *lattice* en una dimensión, en el ejemplo de la tabla el *lattice* evolucionará desde la situación inicial (dos dimensiones) a un *lattice* de una dimensión. De esta manera, en la última fila de la tabla se muestra cuál sería la evolución del vector $\mathbf{y}$ antes de emprender la decodificación del símbolo siguiente. Esta idea comulga más con la filosofía de los detectores SIC y con el concepto de cancelación de interferencias. Por tanto, será desarrollada en la siguiente sección para exponer el algoritmo de proyecciones sucesivas (SPA).

Finalmente, la figura 3.2 muestra gráficamente las equivalencias entre entre el planteamiento a partir de la descomposición ortogonal QR y desde el punto de vista geométrico. El lector puede observar cómo se llega desde uno hasta el otro, a través de sencillas operaciones geométricas: proyecciones, rotaciones y traslaciones.

3.2. SPA Básico. Descripción

En este capítulo ya se ha puesto en evidencia el impacto que la ordenación de las columnas de la matriz de canal tiene sobre la eficiencia computacional del proceso de detección. En los detectores basados en algoritmos de búsqueda en árbol que alcanzan la solución ML, como SD, el deterioro computacional se produce en la tasa de detección, al evaluar las distancias de un número excesivo de nodos. En los detectores que alcanzan soluciones sub-óptimas, como KBest, el impacto es aún mayor ya que éstos recorren el árbol en una única dirección evaluando los nodos a lo ancho del árbol. Aunque en este caso la tasa de detección no se penaliza, si lo hacen las prestaciones BER. Pero donde más crítica es la ordenación es sin duda en los algoritmos SIC, los cuales alcanzan una solución sub-óptima evaluando un único nodo en cada nivel del árbol.

En [2] se demuestra que V-BLAST encuentra la ordenación óptima cuando sólo interviene la matriz de canal, pero el algoritmo general SIC expuesto en la sección 2.3.2 posibilita la aplicación de técnicas de ordenación dependientes no solo de la matriz de canal $\mathbf{H}$, sino además del vector recibido $\mathbf{y}$. Para este segundo caso, se desarrolla el algoritmo SPA [18, 19], el cual encuentra la ordenación óptima teniendo en cuenta la matriz de canal y el vector recibido. Asociado a este proceso de ordenación y siguiendo una metodología similar a la de cualquier detector SIC, los autores de este algoritmo también encuentran una solución sub-óptima que, como se verá en

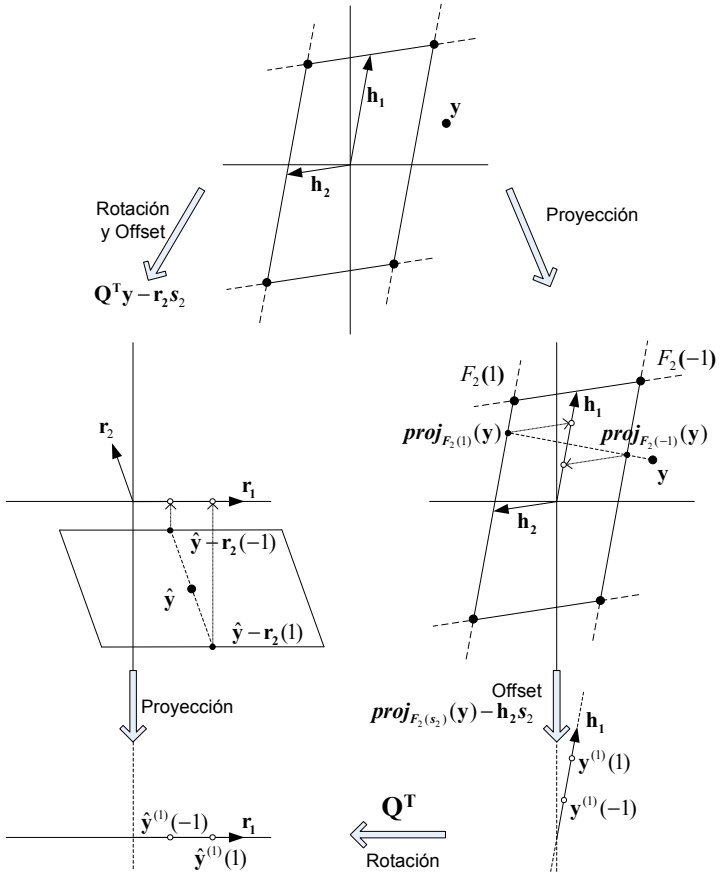


Figura 3.2: Ilustración de las equivalencias entre alternativas de decodificación, para un sistema MIMO 2x2 BPSK.

secciones posteriores, es claramente competitiva con otros detectores.

Considérese el sistema MIMO real representado por la ecuación 2.7 con M=N antenas, siendo $\mathbf{g_i}$ las filas de la matriz $\mathbf{G}$, estimación ZF de $\mathbf{H}$ ($\mathbf{G} = \mathbf{H}^{-1}$). Un detector SIC realizará M veces la secuencia de operaciones descritas en la sección 2.3.2. En el estado k esta secuencia se puede resumir en cuatro puntos:

1. Aplicación de un criterio de ordenación para seleccionar sobre qué fila i_k de la matriz $\mathbf{G}^{(k)}$ se va a realizar la detección.

2. Detección del símbolo s_{i_k}.

3. Cancelación de la influencia del símbolo detectado sobre el vector $\mathbf{y}^{(k)}$.

4. Reducción del *lattice* en una dimensión.

El primer punto de esta secuencia, el algoritmo SPA lo realiza a través de la función *PathSelect*, que se muestra a continuación.

Pseudocódigo 3.1 Función PathSelect $(\mathbf{y}_r^{(k)}, \mathbf{H}^{(k)}, \mathbf{G}^{(k)}, \mathcal{X}, \mathcal{I}^{(k)})$

	for each index i in set $\mathcal{I}^{(k)}$ **do**	
a)	$s_i = \mathbf{g}_{i_k}^{(k)} \cdot \mathbf{y}_r^{(k)}$	Compute estimation
b)	$\alpha_i = \arg\min_{x \in \mathcal{X}} \lvert s_i - x \rvert$	Find nearest set
c)	$\beta_i = \arg\min_{x \in \mathcal{X} \setminus \alpha_i} \lvert s_i - x \rvert$	Second nearest
d)	$\delta_i = d^2(\mathbf{y}_r^{(k)}, F_i(\beta_i))$	Compute *base weight*
	end for	
e)	$i_k = \arg\max_{i \in \mathcal{I}^{(k)}} (\delta_i)$	Apply criterium to select index
f)	$\mathcal{I}^{(k+1)} = \mathcal{I}^{(k)} \setminus i_k$	Remove from index set

La función *PathSelect* ejecuta los siguientes pasos. Primero se computa en a) la estimación en todos los niveles no detectados anteriormente, en b) y c) se localiza el primer (α) y segundo (β) conjunto afín más cercano a esta estimación, los cuales son estrictamente la primera y segunda mejor elección de cada nivel. En d) se calcula la distancia existente entre el vector $\mathbf{y}$ y el segundo conjunto afín más cercano a la estimación lineal. Los autores del algoritmo denominan a esta distancia *base weight* (δ) y demuestran que de entre todas las opciones, la mejor elección es aquel símbolo cuya segunda opción esté más alejada del vector $\mathbf{y}$. Y este es el criterio que se aplica en e) para seleccionar el índice i_k (fila de la matriz $\mathbf{G}$) sobre el que se realizará la detección.

En la tabla 3.2 se muestra con un ejemplo numérico el proceso de detección para un sistema MIMO 2x2 BPSK. La matriz de canal $\mathbf{H}$ y el vector de símbolos recibidos $\mathbf{y}$ se muestran en la ecuación objetivo de la detección ML (2.8), particularizada de la siguiente manera:

$$\hat{\mathbf{s}} = \arg \min_{\mathbf{s} \in \{1,-1\}} \left\| \begin{pmatrix} 9.21 \\ 3.92 \end{pmatrix} - \begin{pmatrix} 1.98 & -5.9 \\ 10.4 & -1.0 \end{pmatrix} \mathbf{s} \right\|^2, \tag{3.6}$$

donde $\hat{s}$ es el vector decodificado.

Sobre el ejemplo mostrado en la tabla cabe destacar que la detección se realiza en dos estados, ya que la dimensión del sistema es $M = 2$.

- En el primer estado $\mathbf{G}^{(1)} = \mathbf{H}^{-1}$ y $\mathbf{y}^{(1)} = \mathbf{y}$ y el símbolo detectado es s_2, ya que su *base weight* (δ_2), distancia entre $\mathbf{y}^{(1)}$ y $\mathcal{F}_2(1)$, es mayor que el *base weight* del nivel 1 (δ_1), distancia entre $\mathbf{y}^{(1)}$ y $\mathcal{F}_1(-1)$.

Tabla 3.2: Ejemplo numérico del algoritmo Basic-SPA numeric example

Estado 1

$\mathbf{G}^{(1)}$	$\begin{pmatrix} -0.0166 & 0.0994 \\ -0.1739 & 0.0331 \end{pmatrix}$
$\mathbf{y}^{(1)}$	$\begin{pmatrix} 9.21 \\ 3.92 \end{pmatrix}$
PathSelect	i s_i α_i β_i δi 1 0.24 1 -1 150.69 **2 -1.47 -1 1 194.97**
Vector Decodificado	$[\hat{s}_1, -1]$

Estado 2

$\mathbf{G}^{(2)}$	$\begin{pmatrix} 0.0177 & 0.0929 \\ 0 & 0 \end{pmatrix}$
$\mathbf{y}^{(2)}$	$\begin{pmatrix} 0.6534 \\ 3.4286 \end{pmatrix}$
PathSelect	i s_i α_i β_i δi **1 0.33 1 -1 50.22**
Vector Decodificado	$[1, -1]$

- Una vez el índice ha sido seleccionado, la detección del símbolo se realiza mediante la asignación $\hat{s}_{ik} = \alpha_{ik}$. En el ejemplo de la tabla $\hat{\mathbf{s}} = [\hat{s}_1, -1]$.
- La detección continúa en el segundo estado una vez que se ha producido la cancelación del símbolo detectado sobre el vector de símbolos recibidos $\mathbf{y}$, y la reducción del *lattice*.

El lector puede observar cómo la segunda fila de $\mathbf{G}$ del ejemplo, es cero. Estas últimas operaciones son realizadas en el algoritmo SPA por la función *TraverseBranch* que se expone a continuación.

Pseudocódigo 3.2 Función TraverseBranch $(\mathbf{y}_r^{(k)}, \mathbf{H}^{(k)}, \mathbf{G}^{(k)}, i_k, \hat{s}_{i_k})$

a) $\mathbf{y}_r^{(k+1)} = proj_{F_{i_k}(\hat{s}_{i_k})}\left(\mathbf{y}_r^{(k)}\right) - \hat{s}_{i_k}\mathbf{h}_{i_k}$ Cancel detected symbol
for each index i in set $\mathcal{I}^{(k)}$ **do**
b) $\mathbf{g}_i^{(k+1)} = proj_{F_{i_k}(0)}(\mathbf{g}_i^{(k)})$ Compute new $\mathbf{G}$
end for

La función *TraverseBranch* ejecuta dos pasos. En a) se realiza la cancelación del símbolo detectado sobre el vector de símbolos recibidos, a través de una proyección sobre el conjunto afín $\mathcal{F}_{i_k}(s_k)$, ($\mathcal{F}_2(-1)$ en el ejemplo). Esta operación ya se vio gráficamente en la sección anterior, en la figura 3.2.

En (b) se computa la nueva matriz estimadora proyectando las filas de $\mathbf{G}^{(1)}$ sobre el conjunto afín $\mathcal{F}_2(0)$, es decir, el correspondiente al índice detectado, pero centrado en el origen. Con estas proyecciones se provoca la anulación de la fila correspondiente al índice i_k detectado (fila 2 en el ejemplo) de $\mathbf{G}^{(1)}$.

Finalmente, el algoritmo SPA [18][19] completo se enuncia de la siguiente manera:

Pseudocódigo 3.3 Algoritmo BasicSPA $(\mathbf{y}_r, \mathbf{H}, \mathcal{X})$

1: $M = rank(\mathbf{H})$ — Set dimension
2: $\mathcal{I}^{(1)} = 1, 2, \ldots, M$ — Initialize index set
3: $\mathbf{y}_r^{(1)} = \mathbf{y}_r$ — Initialize target
4: $\mathbf{G}^{(1)} = \mathbf{H}^{+}$ — Compute ZF estimator
5: **for each** level k from 1 to M **do**
6: $\quad(i_k, \alpha_{i_k}) \leftarrow PathSelect(\mathbf{y}_r^{(k)}, \mathbf{H}, \mathbf{G}^{(k)}, \mathcal{X}, \mathcal{I}^{(k)})$
7: $\quad\hat{s}_{i_k} = \alpha_{i_k}$ — Detect symbol
8: $\quad(\mathbf{y}_r^{(k+1)}, \mathbf{G}^{(k+1)}) \leftarrow TB^{a}(\mathbf{y}_r^{(k)}, \mathbf{H}, \mathbf{G}^{(k)}, i_k, \hat{s}_{i_k})$
9: **end for**
10: **Return** $\hat{\mathbf{s}}$

[a] TraverseBranch

3.3. SPA Extendido (ESPA)

En general los detectores basados en el algoritmo general G-OSIC (Alg. 2.1) son detectores de una sóla pasada, es decir, obtienen una única solución sub-óptima al problema de detección MIMO expresado en (2.3). Los detectores basados en el algoritmo SPA ofrecen como valor añadido la posibilidad de generar de forma iterativa distintas soluciones, según se describe en [19]. Sin embargo, en este trabajo, los autores no establecen ningún método para controlar las repeticiones en las iteraciones, ni se estudia el número de iteraciones requeridas para aproximar la solución ML con un coste computacional razonable. La figura 3.3 muestra las prestaciones BER de un detector SPA Extendido sin ningún control de repeticiones. El detector realiza 11 iteraciones: la básica (BSPA) y 10 extendidas.

El lector puede observar que de las 11 iteraciones realizadas sólo dos son útiles. Por lo tanto, el reto que se aborda en esta sección es encontrar los mecanismos de control de repeticiones necesarios para que este algoritmo sea eficiente y pueda se implementado en *hardware*.

La figura 3.4 muestra con un ejemplo cómo opera el algoritmo SPA-Extendido en el proceso de detección de un sistema MIMO 2x2 16-QAM. En el ejemplo se muestran 3 iteraciones del algoritmo SPA-Extendido, obteniéndose de cada una de ellas un posible candidato (en negrita). El vector

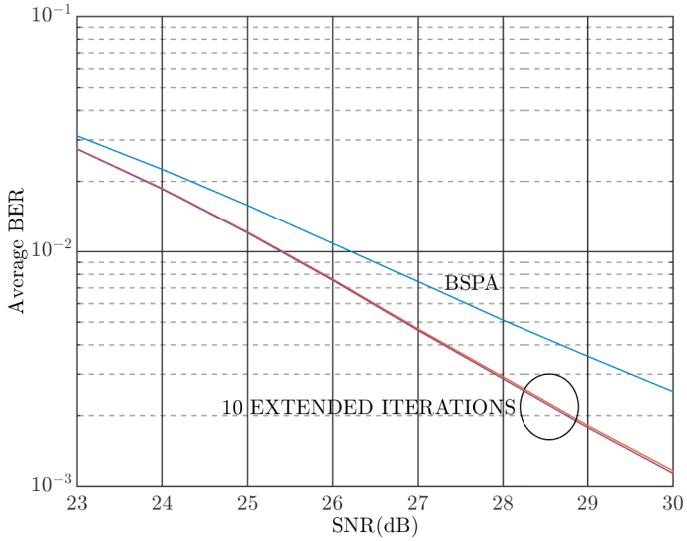


Figura 3.3: Prestaciones BER del detector SPA Extendido sin control de repeticiones

detectado será aquel que satisfaga la ecuación (2.3) de entre todos los candidatos.

Los estados del 1 al 4 corresponden a la primera iteración (desarrollo del algoritmo SPA-Básico). En cada estado se obtiene un *path*, conjunto formado por un índice y su símbolo asociado α, hasta que el vector se completa en el estado 4. Paralelamente, asociado a la obtención de cada componente α se selecciona la segunda mejor opción β y su peso (*base weight*) asociado δ, calculados en las lineas (c) y (d) de la función PathSelect.

Una extensión natural de esta primera pasada (SPA-Básico) es tener en cuenta los ζ mejores caminos que no han sido considerados inicialmente (aquellos β cuyos *base weight* δ son los menores), y por lo tanto, este algoritmo extendido puede alcanzar iterativamente la solución ML. En el ejemplo de la figura 3.4 el menor δ de la primera iteración corresponde al nivel 4 ($\beta = 1,\ \delta = 10.30$), por lo tanto, en el estado 4, además de completarse el vector de la primera iteración, se obtiene el primer *path* de la segunda iteración ($\alpha = 1$ en el nivel 4). Sin embargo, en [19] no se establece ningún método para controlar las repeticiones en las iteraciones, ni se estudia el número de iteraciones requeridas para aproximar la solución ML con un coste computacional razonable. En la figura 3.4 se muestra cómo en ausencia de mecanismos de control de repeticiones en la tercera iteración el vector detectado será inevitablemente igual al de la primera iteración, ya que la primera componente detectada es la misma.

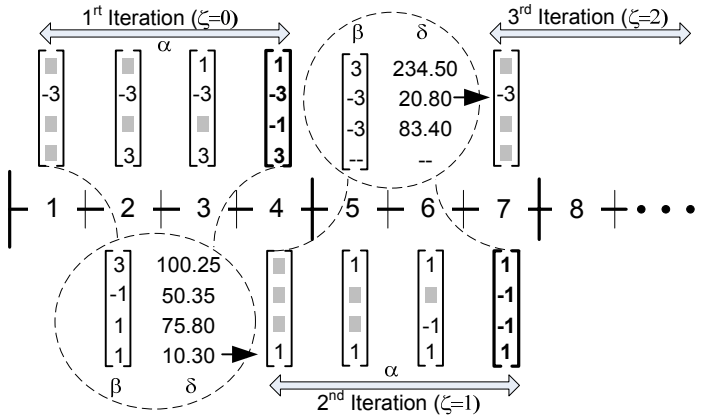


Figura 3.4: Ejemplo de detección del algoritmo SPA-Extendido para un sistema MIMO 2x2 16-QAM.

Aunque en el caso extremo $\zeta \rightarrow \mathcal{B}^{2M_t}$ este detector alcanza la solución ML, nuestras simulaciones demuestran que utilizando sólo las primeras iteraciones el detector consigue una decodificación sub-óptima, de complejidad comparable a la de otros decodificadores con arquitecturas VLSI publicadas.

Para usar el algoritmo SPA iterativamente hay que evitar la repetición de los *'paths'* en las ζ iteraciones consecutivas. Debido a que nuestro objetivo final no es alcanzar la solución ML, ya que su coste computacional sería mucho más alto que el de una búsqueda exhaustiva, no se hace necesario aplicar un control riguroso de las repeticiones, alcanzandose así un equilibrio entre coste computacional y prestaciones BER.

En la siguiente sección se proponen simplificaciones (mecanismos de control de repeticiones) del algoritmo expuesto en [19] para hacer así viable su implementación VLSI.

En la segunda iteración del ejemplo de la figura 2 no aparecen los valores de β y δ del nivel 4. Dado que éste ha sido el nivel elegido como segunda mejor opción de la primera iteración es lógico pensar que en su lugar ahora debería aparecer la tercera mejor opción. Esto implicaría añadir en la función PathSelect dos líneas nuevas similares a la c) y d). El coste computacional de la línea c) (encontrar el tercer elemento) es bajo aún pensando en esquemas de modulación altos como 256-QAM. Sin embargo, el cálculo de un nuevo δ según las ecuaciones (3.5) y (3.4) penaliza de forma excesiva la complejidad computacional y no produce mejoras significativas en las prestaciones BER del detector.

Los mecanismos de control de repeticiones "Table-ESPA" y "Simplified-ESPA" descritos a continuación, ofrecen una alternativa de bajo coste compu-

tacional al cálculo de la tercera mejor ópción.

3.3.1. TESPA: control de repeticiones mediante tabla

El algoritmo Table-SPA muestra el pseudo-código de un detector SPA-Extendido. El algoritmo devuelve la matriz de vectores solución $\hat{\mathbf{S}}$, donde $\mathbf{s}_0$ es la solución correspondiente al SPA-Básico, y el resto las de las sucesivas iteraciones ζ.

El método utilizado para controlar las repeticiones es una tabla donde en cada fila se almacena información relativa a cada iteración (el primer *path* tomado y su δ asociado). Este proceso se realiza de forma continua de forma que al final de cada iteración ya está disponible el primer *path* de la siguiente (función GetPath en la línea 13 del algoritmo TableSPA). En la línea 5 se inicializa la tabla de $(\zeta + 1)$ filas y 2 columnas estableciendo a ∞ los pesos δ (entradas de la columna 2).

Pseudocódigo 3.4 Función SetPath $(\mathcal{PT}, iter, NewPath, \delta_{new})$

a) $\delta = \mathcal{PT}(iter, 2)$	Read old δ
b) **if** $(\delta_{new} \leq \delta)$ **and** $(NewPath \notin \mathcal{PT})$	Validate New *path*
c) $\quad \mathcal{PT}(iter, 1) \leftarrow NewPath$	Write new *path*
d) $\quad \mathcal{PT}(iter, 2) \leftarrow \delta_{new}$	Write new δ
e) **end if**	
f) **Return** $\mathcal{PT}$	

La función SetPath es la responsable de actualizar la tabla. Para actualizar una fila se tienen que cumplir simultáneamente dos condiciones:

- Que el δ asociado al *path* sea menor que el anterior almacenado.
- Que el *path* sea nuevo (distinto a los tomados en las iteraciones anteriores).

En la línea 7 del algoritmo TableSPA se establece el *path* de la primera iteración y en la línea 8 el primer candidato para la segunda iteración, el resto de candidatos se comprueban en la línea 18 a medida que se van obteniendo componentes vectoriales.

En el ejemplo de la figura 3.4 la repetición que se produce en la tercera iteración no se hubiera producido de haberse utilizado este mecanismo de control. La tabla 3.3 muestra cómo sería la tabla de *paths* al comienzo de la tercera iteración del ejemplo. El *path* (2,-3) con $\delta = 20.80$, ha sido rechazado aún teniendo el δ mas pequeño por estar ya en la tabla (iteración $\zeta = 0$). El *path* tomado en su lugar es el siguiente no tomado anteriormente, con δ más pequeño (3,-3).

Este método no garantiza al completo la no repetición de candidatos, ya que el hecho de que las dos primeras componentes vectoriales sean distintas

Pseudocódigo 3.5 Algoritmo Table-ESPA $(\mathbf{y}_r, \mathbf{H}, \mathcal{X}, \zeta)$

```
M = rank(H)                                                  Set dimension
I^(0) = 1, 2, ..., M                                        Initialize index set
y_r^(0) = y_r                                               Initialize target
G^(0) = H^+                                                 Compute ZF estimator
PT ← InitPathTable                                          Initialize path table
(i_1, α_i1, β_i1, δ_i1) ← PathSelect(y_r^(0), H, G^(0), X, I^(0))
PT ← SetPath(PT, 0, [i_1, α_i1], δ_i1)
PT ← SetPath(PT, 1, [i_1, β_i1], δ_i1)
for each iteration j from 0 to ζ do
  I^(1) = I^(0)
  y_r^(1) = y_r^(0)
  G^(1) = G^(0)
  [i_sel, γ_sel] ← GetPath(PT, j)
  ŝ_{i_sel,j} = γ_sel                                       Detect symbol
  for each level k from 1 to M − 1 do
    (y_r^(k+1), G^(k+1), I^(k+1)) ← TB(y_r^(k), H, G^(k), i_sel, γ_sel, I^(k))
    (i_k, α_ik, β_ik, δ_ik) ← PathSelect(y_r^(k), H, G^(k), X, I^(k))
    PT ← SetPath(PT, j + 1, [i_k, β_ik], δ_ik)
    i_sel = i_k
    γ_sel = α_ik
    ŝ_{i_sel,j} = γ_sel                                     Detect symbol
  end for
end for
Return Ŝ
```

no implica que al final los dos vectores sean distintos, i.e el vector [1,-3,-1,3] puede empezar a dibujarse desde el *path* (2,-3) ó desde el (4,3) y en ambos casos se podría llegar a él.

En la sección 3.5 se realiza un análisis exhaustivo de las prestaciones alcanzadas por este detector mediante simulación, y un estudio de la viabilidad de su implementación *hardware*.

Tabla 3.3: Tabla de *paths* correspondiente al ejemplo de la figura 3.4.

Iter (ζ)	**Path**	**Base Weight**(δ)
0	(2,-3)	50.35
1	(4,1)	10.30
2	(3,-3)	80.40

3.3.2. SESPA: control simplificado de repeticiones

Aunque el uso de la función *SetPath* para el control de repeticiones TableSPA descrito anteriormente no penaliza la tasa de detección (*throughput*) del detector, resulta evidente que el mantenimiento de la tabla de *'paths'* pa-

ra un número de iteraciones cercanas a 8 sí incrementará el área del detector. El algoritmo Simplificado-ESPA utiliza un sencillo control de repeticiones con un coste computacional muy bajo.

Pseudocódigo 3.6 Algoritmo Simplified-ESPA($\mathbf{y}_r, \mathbf{H}, \mathcal{X}, \zeta$)

In Algorithm 3.5, delete lines 5,7,8 and 18 and replace line 13 by:

13.1: **if** $j = 0$ **then**
13.2: $\gamma_{sel} = \alpha_{i_{sel}}$ Nearest in first iteration
13.3: **else**
13.4: $\gamma_{sel} = \beta_{i_{sel}}$ 2^{o} nearest in other iteration
13.5: **end if**
13.6: $i_{sel} = \arg\min_{i \in \mathcal{I}^{(1)}} \delta_i$
13.7: $\delta_{sel} = 2^{10}$

Esta modificación supone una relajación del control de repeticiones, eliminando todas las referencias a la tabla de *'paths'*. El primer índice de cada iteración se selecciona buscando simplemente el *base weight* mínimo y una vez se ha seleccionado el índice se le asigna un valor alto a su *base weight* asociado. Esto garantiza que en la siguiente iteración este *path* no será seleccionado.

En el ejemplo de la figura 3.4 en la segunda iteración el símbolo en blanco β será 1, como en la primera iteración, y su δ correspondiente será sobrescrito a 2^{10}. De esta manera el *path* tomado en la segunda iteración no se repite en la tercera. El lector puede observar que este mecanismo controla las repeticiones en iteraciones contiguas, no así en las alternas.

En la sección 3.5 se realiza una evaluación exhaustiva de las prestaciones alcanzadas por este detector mediante simulación, y un estudio de viabilidad para su posible implementación *hardware*.

3.4. Generación de las salidas *hard output* y *soft output*

El objetivo perseguido por la detección MIMO es encontrar las salidas *hard output* y/o *soft output*, a través de las ecuaciones que se describen en la sección 2.2.1. Aunque los dos modelos de detección son de naturaleza diferente, tienen en común que para la obtención de cualquiera de las dos salidas es necesario calcular la distancia euclídea entre el vector de símbolos recibidos $\mathbf{y}$ y $\mathbf{Hs}$, vector detectado transformado por la matriz de canal. Esta distancia se calcula mediante la ecuación

$$d(\mathbf{s}) = \|\mathbf{y_r} - \mathbf{Hs}\|^2. \tag{3.7}$$

Los algoritmos TESPA (3.5) y SESPA (3.6), descritos en la sección 3.3,

muestran el pseudo-código de un detector SPA-Extendido que devuelve el conjunto de vectores solución $\hat{\mathbf{S}}$, donde $\hat{\mathbf{s}}_0$ es la solución correspondiente al SPA-Básico, y el resto las de las sucesivas iteraciones ζ. La distancia $\hat{\mathbf{s}}_j$ asociada a cada uno de ellos se calcula aplicando la ecuación 3.7.

1. **La detección *hard output* (ML)** tiene como objetivo proporcionar la estimación "hard"de cada bit, a través de la ecuación

$$\hat{\mathbf{s}} = \arg \min_{\hat{\mathbf{s}}_\mathbf{j} \in \hat{\mathbf{S}}} d(\hat{\mathbf{s}}_j), \tag{3.8}$$

   donde $\hat{\mathbf{S}}$ es el conjunto de vectores solución obtenidos de las diferentes iteraciones de los algoritmos ESPA.

   La solución *hard output* del detector será aquella que de entre todas satisfaga la ecuación 3.8.

2. **La detección soft output** tiene como objetivo encontrar la información APP de cada bit utilizando el valor "*Log-Likelihood Ratio*" (LLR), cuya simplificación, apta para la implementación *hardware*, propuesta en [23][24], está representada por las ecuaciones (2.5 y 2.9).

   Los valores L de cada bit se calculan según la siguiente expresión:

$$L\left(b_{l,k}|\mathbf{y_r}\right) \approx (\Lambda_{l,k}^{-1} - \Lambda_{l,k}^{+1}), \tag{3.9}$$

   donde $b_{l,k}$ representa el k-ésimo bit del l-ésimo símbolo y $\Lambda_j = \dfrac{d(\hat{\mathbf{s}}_j)}{N_0}$. Para el caso particular de un sistema MIMO 4x4 64-QAM el LLR será un vector de 24 posiciones (4 símbolos x 6 bits/símbolo). Los valores asignados a estas posiciones son los Λ_j asociadas a los candidatos $\hat{\mathbf{s}}_j$ obtenidos en cada iteración. La asignación de estos valores se realiza según los siguientes criterios:

   - Dado el vector de símbolos $\hat{\mathbf{s}}_j$, candidato j, y su métrica asociada $d(\hat{\mathbf{s}}_j)$. El valor Λ_j será escrito en las posiciones correspondientes a los bits asociados a los símbolos del vector.
   - Se escribirá el valor Λ_j en su posición correspondiente, +1 o -1, cuando su valor sea mejor que el anterior almacenado (menor métrica).

   Una vez procesados todos los vectores candidatos, puede ocurrir que alguna posición del vector LLR se haya quedado sin escribir. En la literatura especializada se proponen diversas soluciones a este problema [23][35][16].

Pseudocódigo 3.7 Algoritmo Output $(\mathbf{y_r}, \mathbf{H}, \hat{\mathbf{S}}, Soft/Hard)$

for each symbol vector $\hat{\mathbf{s}}_j \in \hat{\mathbf{S}}$ **do**
  $d(\hat{\mathbf{s}}_\mathbf{j}) = \|\mathbf{y_r} - \mathbf{H}\hat{\mathbf{s}}_\mathbf{j}\|^2$ — Compute distances
end for
************Hard Output*************
case $Hard$ — *Hard output*
  $\hat{\mathbf{s}} = \arg\min_{\hat{\mathbf{s}}_\mathbf{j} \in \hat{\mathbf{S}}} d(\hat{\mathbf{s}}_j)$
  Return $\hat{\mathbf{s}}$ — Return *hard output*
end case
************Soft Output*************
case $Soft$ — *Soft output*
  $dim_{LLR} = M_t \times log_2(q)$ — MIMO $(M_t \times N_r)$ $q-QAM$
  Initialize L^{+1}, L^{-1} and LLR
  for each index i from 1 to $\dim_{LLR}$ **do**
    $L_i^{+1} = \phi$
    $L_i^{-1} = \phi$
    $LLR_i = \phi$
  end for
  Set L values
  for each symbol vector $\hat{\mathbf{s}}_j \in \hat{\mathbf{S}}$ **do**
    for each symbol $\hat{s}_{j,l} \in \hat{\mathbf{s}}_j$ **do**
      for each bit $b_{j,l,k} \in \hat{s}_{j,l}$ **do**
        if $\left(b_{j,l,k} == 1\right)$ **and** $\left(L_i^{+1} > d(\hat{\mathbf{s}}_j)\right)$ **then** $L_i^{+1} = d(\hat{\mathbf{s}}_j)$
        if $\left(b_{j,l,k} == -1\right)$ **and** $\left(L_i^{-1} > d(\hat{\mathbf{s}}_j)\right)$ **then** $L_i^{-1} = d(\hat{\mathbf{s}}_j)$
      end for
    end for
  end for
  Set empty L to L_{const} and compute LLR
  for each index i from 1 to $\dim_{LLR}$ **do**
    if $\left(L_i^{+1} = \phi\right)$ **then** $L_i^{+1} = L_{const}$
    if $\left(L_i^{-1} = \phi\right)$ **then** $L_i^{-1} = L_{const}$
    $LLR_i = \left(L_i^{-1} - L_i^{+1}\right)/N_0$ — Compute LLR
  end for
  Return LLR — Return *soft output*
end case

a) La asignación del valor $|\Lambda_{min} - \Lambda_{max}|$ en las posiciones vacías, es decir el módulo de la diferencia entre la mejor y la peor métrica.

b) La asignación de un valor extremo, Λ_{extr}, en función del radio de búsqueda del algoritmo *"Sphere Decoding"*.

c) La asignación de un valor empírico, Λ_{const}, obtenido mediante simulaciones previas, que optimiza las prestaciones FER del decodificador.

En este trabajo se ha seleccionado la última opción ya que los resultados obtenidos, a través de esta, son manifiestamente mejores.

El algoritmo 3.7 describe el pseudocódigo de las operaciones necesarias para obtener las salidas *hard output* y *soft output*.

3.5. Resultados en precisión infinita

Los algoritmos estudiados en las secciones anteriores requieren ser evaluados en cuanto a prestaciones BER y complejidad computacional se refiere. En esta sección se presenta, como contribución de esta tesis, un estudio exhaustivo de la evaluación realizada de los mismos. De las conclusiones extraídas de este estudio podrá determinarse la viabilidad, de la implementación *hardware*, de los diferentes detectores presentados.

Se han obtenido resultados de los dos modelos de detección descritos en la sección 2.2.1, *hard output* y *soft output*, mediante simulaciones a través del método de Motecarlo en coma flotante, es decir, sin imponer restricciones a la precisión de los datos. Aunque el método de simulación empleado para ambos modelos de detección es el mismo, Montecarlo, las diferencias en la formulación de ambos modelos provocan, a su vez, diferencias en el planteamiento de las simulaciones.

- Para la obtención del *hard output* se ha simulado un sistema de comunicación MIMO sin corrección de errores. El parámetro de calidad obtenido es la tasa de error de bit BER (Bit Error Rate) en función de la SNR del sistema.
- Para la obtención del *soft output* se ha definido un sistema de comunicación MIMO basado en corrección de errores (FEC), mediante un codificador LDPC. El parámetro de calidad obtenido es la tasa de error de tramas FER (Frame Error Rate) en función de la SNR del sistema.

En las secciones posteriores se describen con detalle los sistemas de comunicación simulados. La sección 3.5.1 se centra en presentar los resultados

BER de los detectores ESPA, desarrollados en este capítulo, mientras que las secciones posteriores, 3.5.2 y 3.5.3, abordan la comparativa, del *hard output* y del *soft output* respectivamente, con otros detectores KBest publicados en la literatura especializada.

3.5.1. Análisis de prestaciones BER de detectores ESPA

En esta sección se presentan los resultados de prestaciones BER alcanzados por los algoritmos estudiados en este capítulo. Estos resultados se han obtenido mediante simulaciones a través del método de Montecarlo, promediando las curvas BER obtenidas en diferentes realizaciones de canal. En particular cada curva de esta sección ha sido obtenida promediando 10.000 canales aleatorios. Se han utilizado tramas de 100 símbolos y el sistema se ha modelado según se indica en la sección 2.1 para sistemas MIMO 4x4 64QAM y 256QAM.

- **Basic-SPA:** La figura 3.5 muestra una compartiva de prestaciones BER entre el algoritmo SPA-Básico descrito en la sección 3.2, el decodificador VBLAST y el Maximum Likelihood (ML). Se observa una mejora notable con respecto al detector VBLAST, debido a que el algoritmo SPA-Básico encuentra la ordenación óptima de la matriz de canal teniendo en cuenta el vector recibido, según se explica en la sección 3.2.
- **Table-Extended-SPA:** El detector TESPA realiza el control de las repeticiones mediante una tabla según se explica en la sección 3.3.1. La figura 3.6 muestra 25 iteraciones realizadas con este detector. Entre el decodificador VBLAST y el Maximum Likelihood (ML) se puede observar cómo a medida que aumenta el número de iteraciones del decodificador TESPA sus prestaciones BER se aproximan a las ML. Se observa una mejoría clara en las prestaciones BER del detector en las primeras 8 iteraciones mientras que a partir de la octava se ralentiza la convergencia con la curva ML.
- **Simplified-Extended-SPA:** Las prestaciones BER del detector sin control de repeticiones SESPA descrito en la sección 3.3.2 se muestran en la figura 3.7. Entre el decodificador VBLAST y el Maximum Likelihood (ML) se puede observar cómo a medida que aumenta el número de iteraciones del decodificador SESPA sus prestaciones BER se aproximan a las ML, hasta que se produce un estancamiento aproximadamente a partir de la 5 iteración.
- **TESPA vs. SESPA:** Es importante hacer notar que los detectores TESPA y SESPA no se comportan de igual manera en las primeras

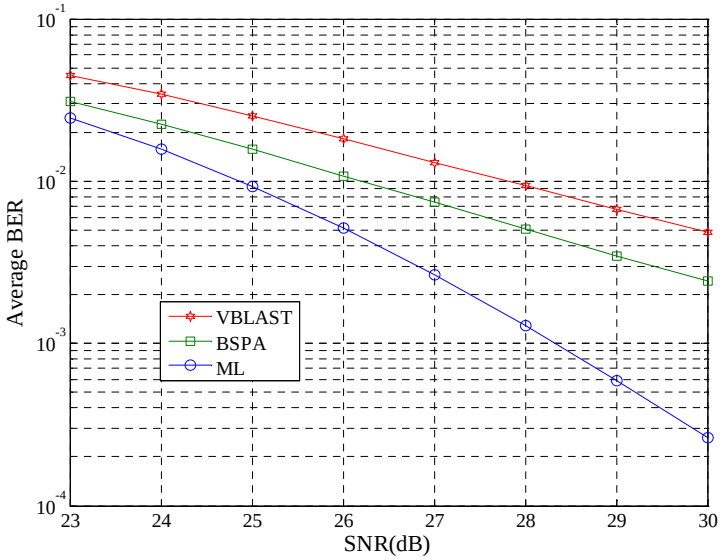


Figura 3.5: Comparativa de prestaciones BER de detectores MIMO 4x4 64-QAM. Basic-SPA

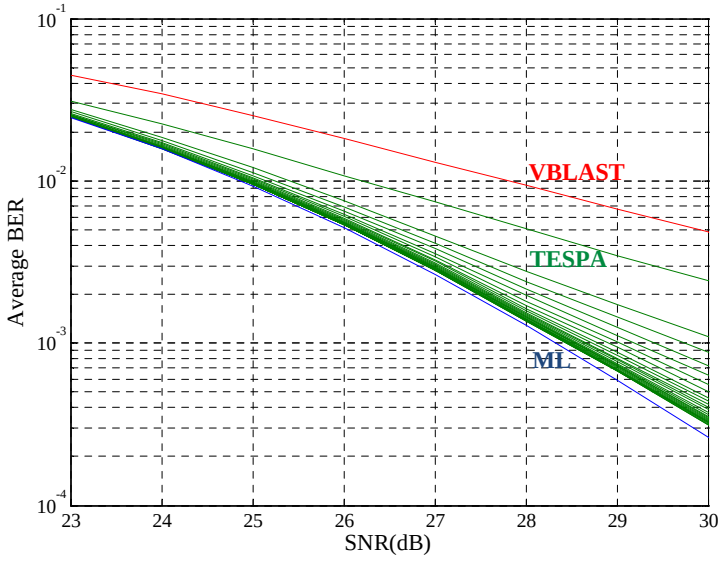


Figura 3.6: Evolución de prestaciones BER del detector MIMO 4x4 64-QAM Table-ESPA en función del número de iteraciones programadas

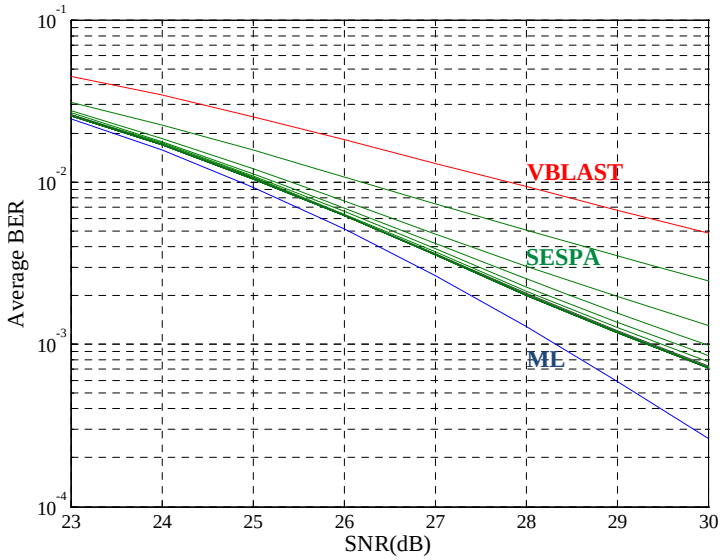


Figura 3.7: Evolución de prestaciones BER del detector MIMO 4x4 64-QAM Simplified-ESPA en función del número de iteraciones programadas

iteraciones. Aunque pueda parecer que las prestaciones BER de ambos son idénticas en las primeras iteraciones, no es así. De hecho sólo coinciden en la primera iteración (BSPA). La figura 3.8 muestra la diferencia que se alcanza en la cuarta iteración entre las prestaciones BER de ambos decodificadores. En este punto el detector SESPA prácticamente ha saturado en cuanto a prestaciones BER se refiere, mientras que al decodificador SESPA aún le queda un recorrido notable, como se observa en las figuras 3.6 y 3.7.

3.5.2. Comparativa *hard output* entre detectores ESPA y KBest

En esta sección se aborda de forma exhaustiva un estudio comparativo entre detectores ESPA y KBest *hard output*, obtenida esta salida según el algorimo 3.7. En particular, se establecen comparaciones en cuanto a prestaciones BER y complejidad computacional de ambos detectores, como punto de partida para determinar la viabilidad de implementación *hardware* del detector ESPA.

La figura 3.9 muestra las prestaciones BER alcanzadas por lo detectores TESPA y SESPA para un sistema MIMO 4x4 64-QAM. Como ya se ha visto

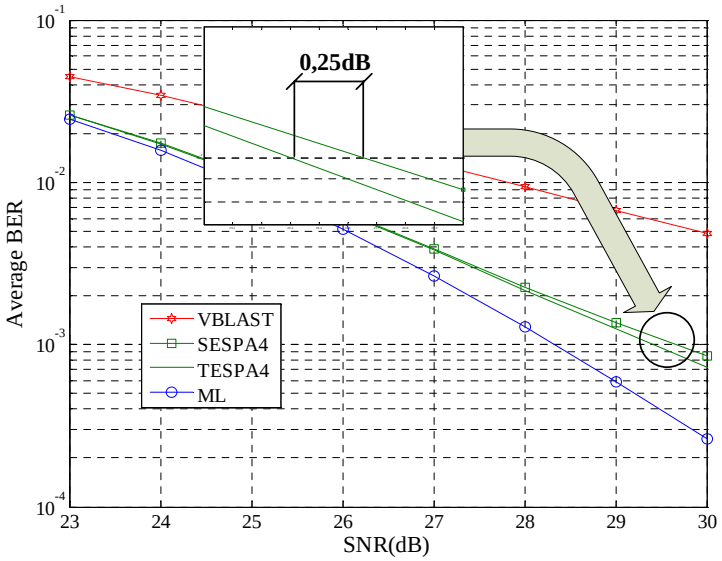


Figura 3.8: Comparativa de prestaciones BER de detectores MIMO 4x4 64-QAM. Simplified-ESPA y Table-ESPA con 4 iteraciones

en la sección anterior, las curvas se mueven entre las prestaciones BER de los detectores VBLAST y ML, y el detector Basic-SPA (BSPA) corresponde a la primera iteración de sendos detectores. A medida que aumenta el número de iteraciones las curvas BER se aproximan a la del detector ML. En el caso del detector SESPA esta mejora se detiene en la quinta iteración y, por lo tanto, se hace necesario utilizar el mecanismo de control de repeticiones que implementa el detector TESPA para que continue la mejora. La figura 3.10 muestra resultados similares para un sistema MIMO 4x4 256-QAM.

Para poder realizar una comparación objetiva de las prestaciones BER alcanzadas y de la complejidad computacional de los detectores implementados en este trabajo, se ha seleccionado un detector KBest cuya implementación VLSI ha sido desarrollada y presentada en la bibliografía especializada [16].

Los autores de este trabajo presentan la implementación VLSI de un detector basado en el algoritmo KBest para un sistema MIMO 4x4 64-QAM. Para alcanzar una tasa de detección óptima la selección de candidatos se realiza según un radio de búsqueda, lo que provoca que la complejidad sea dependiente de la SNR del sistema y se emplea un mecanismo de ordenación relajado de las listas de candidatos, que reduce la tasa de detección, pero como contrapartida empeora las prestaciones BER alcanzadas.

En las figuras 3.9 y 3.10 se muestran las prestaciones BER correspondien-

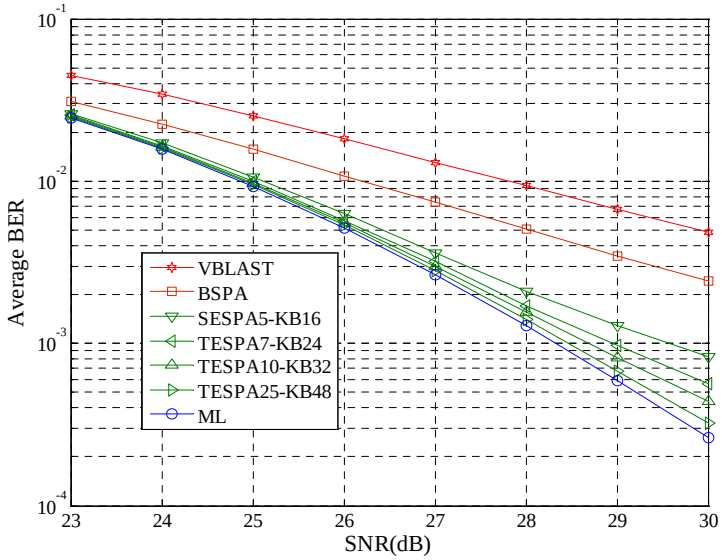


Figura 3.9: Comparativa de prestaciones BER de detectores MIMO 4x4 64-QAM. SPA y KBest

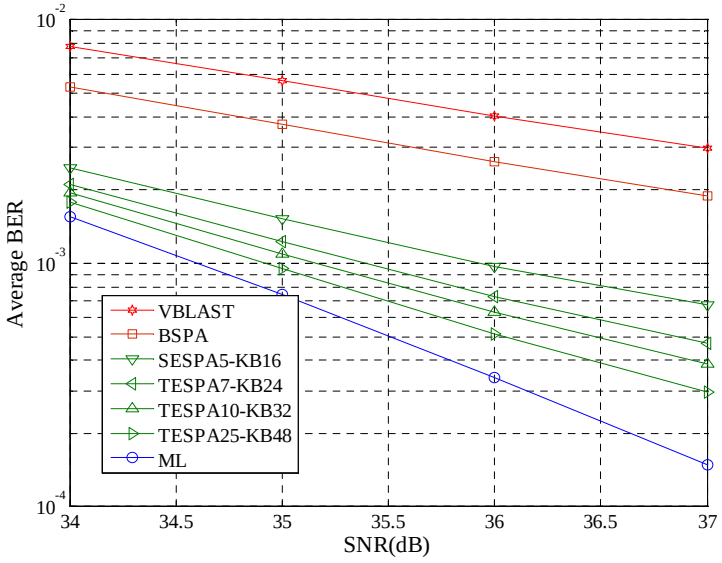


Figura 3.10: Comparativa de prestaciones BER de detectores MIMO 4x4 256-QAM. SPA y KBest

Tabla 3.4: Equivalencias entre detectores ESPA y KBest

ESPA-(Nº Iter)	KBest-(K)
SESPA-5	KB-16
TESPA-7	KB-24
TESPA-10	KB-32
TESPA-25	KB-48

$\triangle SNR \leq 0.1dB\ para\ BER = 10^{-3}$

tes a detectores ESPA y KBest equivalentes: dos detectores se consideran equivalentes si alcanzan la BER de 10^{-3} con una diferencia de SNR menor de 0.1 dB. Por ejemplo un TESPA con 10 iteraciones es equivalente a un KBest con $K = 32$, como se muestra en la tabla 3.4.

La figura 3.11 muestra la carga computacional de las parejas de detectores equivalentes para sistemas MIMO 4x4 64-QAM y 256-QAM. La carga computacional se ha medido en términos de productos, sumas y comparaciones. Como la complejidad computacional del detector KBest seleccionado para la comparación depende de la SNR del canal, las medidas han sido tomadas a la SNR cuya BER es de 10^{-3}. En el caso de los detectores ESPA evaluados la complejidad computacional es fija, independiente del canal y únicamente dependiente del número de iteraciones programadas.

La figura muestra que SESPA con 5 iteraciones (S5) y TESPA con 7 (T7) o 10 (T10) iteraciones son más eficientes que sus equivalentes KBest para 64-QAM (KB16, KB24 y KB32). Sólamente el detector KB48 ($K = 48$) tiene menor complejidad que su equivalente TESPA con 25 iteraciones (T25), pero como se puede apreciar en la figura 3.9 la mejora introducida por el detector T25 sobre el T10 es de apenas 0.25 dB a 10^{-3}, por lo tanto su utilización es, cuanto menos, discutible.

Si embargo, para una constelación de nivel superior, tal como 256-QAM, los detectores ESPA tienen siempre una complejidad computacional claramente inferior a los detectores KBest. Por ejemplo, para 64-QAM el detector TESPA con 10 iteraciones tiene una reducción de complejidad de 32 % en productos, 24.5 % en sumas y 47.5 % en comparaciones respecto a su equivalente KBest (KB32), mientras que en 256-QAM la diferencia entre los mismos detectores es más favorable al ESPA: 82 % en productos, 77.5 % en sumas y 60 % en comparaciones.

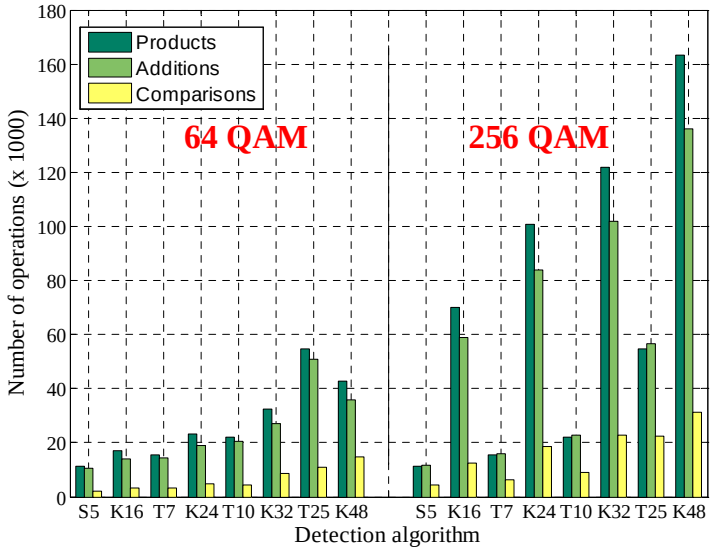


Figura 3.11: Comparativa de complejidad de detectores ESPA Y KBest para sistemas MIMO 4x4 64QAM y 256QAM

3.5.3. Comparativa *soft output* entre detectores ESPA y KBest

Las prestaciones *soft output* de los detectores propuestos han sido evaluadas mediante simulaciones considerando un sistema MIMO 4x4 64-QAM OFDM con FFT de 64 puntos y con codificación de canal LDPC. De las 64 subportadoras, 48 son portadoras de datos y el resto se utilizan para pilotos y portadoras virtuales según se define en el estándar 802.11n. Cada subportadora de una canal MIMO permanece constante durante la transmisión de una trama completa y es "flat fading"(todas las entradas de la matriz de canal MIMO son variables aleatorias, gausianas e independientes). El código LDPC se ha seleccionado con un ratio de 2/3 y una longitud de trama 1944 bits y realiza 20 iteraciones. No hay iteración entre el detector y el decodificador. La SNR del canal MIMO se ha definido como en [23], siendo R la tasa de codificación de canal ($R = 1$ sistema sin codificar), la SNR se define como

$$\frac{E_b}{N_o}(dB) = \frac{E_s}{N_o}(dB) + 10\log_{10}\left(\frac{N_r}{R \cdot M_t \cdot b}\right), \tag{3.10}$$

donde E_s es la energía media de símbolo de la constelación QAM.

Con estas condiciones, se han simulado los detectores SESPA, TESPA y KBest para obtener, a través del método de Montecarlo, la tasa de error de

trama (FER) promediada para diferentes canales. Además, como referencia para establecer las comparaciones se ha simulado, también, un detector "List Sphere Decoding" (LSD), que obtiene una lista con los mejores candidatos, aquellos que cumplen la ecuación 2.3 con menor distancia, entre los que se encuentra, obviamente, la solución óptima ML.

El LLR necesario para la corrección de errores se ha obtenido a través del algoritmo 3.7 descrito en la sección 3.4.

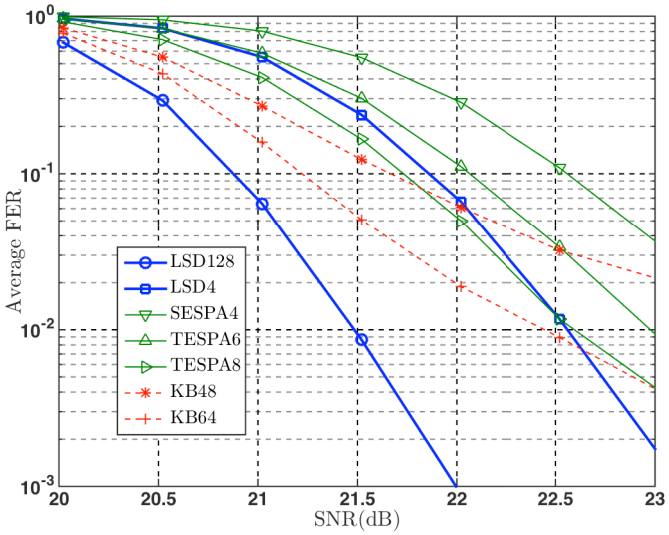


Figura 3.12: Comparativa de prestaciones FER de detectores ESPA Y KBest para sistemas MIMO 4x4 64QAM

La figura 3.12 muestra los resultados obtenidos. Las mejores prestaciones FER son obtenidas por el detector elegido como referencia, LSD128. Este detector, basado en *"Sphere Decoding"*, a diferencia de los otros, alcanza la solución ML y con ella la mejor lista de candidatos, es decir aquellos cuyas métricas más se aproximan a la solución ML. En otro extremo están los detectores KBest y ESPA, que no alcanzan la solución ML y ofrecen menos candidatos para el cálculo del LLR.

Resulta sorprendente el resultado obtenido por detector ESPA8, que con sólo 8 candidatos prácticamente iguala al detector KB48(48 candidatos) a la tasa de error 10^{-1} y al detector KB64(64 candidatos) a la tasa $4 \cdot 10^{-3}$. La explicación de hecho reside en que para el cálculo del LLR no sólo es importante la cantidad de candidatos sino, además, la calidad de los mismos. Los detectores KBest son detectores de búsqueda en árbol de una sola pasada, lo que provoca que los candidatos generados tengan una alta correlación entre ellos, es decir, muchos símbolos parecidos. Por lo tanto, a

la hora de calcular el LLR quedan muchas posiciones vacías, que hay que completar utilizando las técnicas descritas anteriormente. Esto supone un empobrecimiento de la información APP que recibe el sistema de corrección de errores.

3.6. Conclusiones

El algoritmo general SIC (Alg. 2.1) expuesto en la sección 2.3.2 posibilita la aplicación de técnicas de ordenación dependientes no solo de la matriz de canal $\mathbf{H}$, sino además del vector recibido $\mathbf{y}$. Para este segundo caso, se desarrolla el algoritmo SPA [18, 19], el cual encuentra la ordenación óptima teniendo en cuenta la matriz de canal y el vector recibido. Asociado a este proceso de ordenación y siguiendo una metodología similar a la de cualquier detector SIC, los autores de este algoritmo también encuentran una solución sub-óptima.

En general los detectores basados en el algoritmo general G-OSIC (Alg. 2.1) son detectores de una sóla pasada, es decir, obtienen una única solución sub-óptima al problema de detección MIMO expresado en (2.3). Los detectores basados en el algoritmo SPA ofrecen como valor añadido la posibilidad de generar de forma iterativa distintas soluciones a través del algoritmo ESPA (Extended SPA), según se describe en [19]. Sin embargo, en este trabajo, los autores no establecen ningún método para controlar las repeticiones en las iteraciones, ni se estudia el número de iteraciones requeridas para aproximar la solución ML con un coste computacional razonable.

Como contribución de esta tesis se presentan en la sección 3.3 de este capítulo dos mecanismos que permiten el control de las repeticiones en las iteraciones sucesivas del algoritmo ESPA así como el método adaptado para la obtención de un LLR válido para generar la salida *soft output*.

1. TESPA (*Table-ESPA*), utiliza una tabla para almacenar el primer símbolo de cada iteración. Con este mecanismo se reduce la posibilidad de repetición pero no se elimina totalmente.

2. SESPA (*Simplified-ESPA*), es un mecanismo simplificado de repetición que evita repeticiones entre una iteración y la siguiente.

Los detectores basados en estos dos mecanismos han sido evaluados en la sección 3.5 en cuanto a prestaciones BER y complejidad computacional del *hard output*, y prestaciones FER del *soft output*, y han sido comparados con otros detectores publicados en la literatura especializada obteniendo las siguientes conclusiones:

1. Las prestaciones BER obtenidas por el detector BSPA (SPA-Básico)

mejoran notablemente las del detector VBLAST, de menor complejidad computacional.

2. El detector TESPA mejora claramente sus prestaciones BER en las primeras 8 iteraciones mientras que a partir de la octava se ralentiza la convergencia con la curva ML.

3. El detector SESPA mejora claramente sus prestaciones BER en las primeras 5 iteraciones mientras que a partir de la quinta se ralentiza la convergencia con la curva ML.

4. Los detectores basados en ESPA igualan las prestaciones BER de los detectores KBest, siendo especialmente competitivos, en cuanto a complejidad computacional se refiere, cuando crece el esquema de modulación utilizado, en particular para el caso estudiado 256-QAM.

5. Los detectores basados en ESPA alcanzan unas prestaciones FER (*soft output*) claramente competitivas con los detectores KBest, debido a la mayor calidad del LLR generado por el ESPA.

Los detectores KBest han sido ampliamente utilizados en el literatura especializada para su implementación *hardware*, debido a que con ellos se consigue el necesario equilibrio entre prestaciones BER y complejidad computacional que hace que una arquitectura VLSI sea viable.

A la vista de los resultados expuestos en estas conclusiones los detectores basados en el algoritmo ESPA se muestran como firmes candidatos para su implementación *hardware* ya que el balance entre prestaciones BER y complejidad computacional que presentan es altamente competitivo con los detectores KBest y, además, presentan como ventajas adicionales su viabilidad, en cuanto a complejidad computacional se refiere, para 256-QAM (recientemente adoptado por el estándar 3GPP-LTE) y su capacidad para generar un salida *soft output* de calidad.

Por lo tanto, el reto que se aborda en el siguiente capítulo es el diseño de una arquitectura VLSI para los detectores SESPA y TESPA que han sido presentados y evaluados en este capítulo.

Capítulo 4

Implementación del algoritmo SPA

Los resultados obtenidos a partir de las simulaciones realizadas, que se han expuesto en el capítulo anterior, no sólo muestran que el algoritmo SPA es claramente competitivo en lo que respecta al binomio prestaciones/complejidad, sino que además, permite ser configurado para adaptarse a diferentes sistemas MIMO en lo que respecta a antenas y esquemas de modulación.

Del estudio del número de iteraciones útiles de los algoritmos *Extended-SPA* (ESPA), también realizado en el capítulo anterior, se desprende que, en el mejor de los casos, es decir, para el algoritmo TESPA, la obtención de más de 8 candidatos no provoca una mejoría de prestaciones BER considerable. Por este motivo, y teniendo en cuenta el coste computacional de cada iteración, se establece como parámetro de diseño de la arquitectura *hardware* la configuración del número de candidatos a obtener, siendo 8 el máximo.

En este capítulo se presenta una arquitectura *hardware* apta para implementar el algoritmo Table-ESPA y una variante de menor complejidad para el algoritmo Simplified-ESPA. Los objetivos perseguidos por este capítulo se resumen básicamente en cuatro puntos:

- Detallar la arquitectura propuesta para el desarrollo de los algoritmos SPA.
- Mostar los resultados de implementación en tecnología FPGA y ASIC

CMOS 90 nm.

- Realizar un estudio de las posibilidades de paralelización/serialización que ofrece la arquitectura diseñada.
- Comparar la implementación ASIC con otras respecto a diferentes parámetros característicos del Hardware.

Para ello, en las tres primeras secciones se desarrolla la arquitectura empezando desde el nivel superior en la sección 4.1 y descendiendo a niveles mas bajos en las secciones 4.2 y 4.3. En las secciones 4.4 y 4.5 se muestran los resultados de implementación y las posibilidades de paralelización de la arquitectura respectivamente, y finalmente en la sección 4.6 se realiza una comparación exhaustiva con otros detectores publicados en la literatura especializada antes de exponer las conclusiones.

4.1. Descripción general de la arquitectura

La figura 4.1 muestra el diagrama general del detector MIMO SPA propuesto. Se compone de dos grandes bloques: el bloque de detección de símbolos y el bloque de procesado de salida. Este detector es apto para decodificar sistemas MIMO desde 2x2 hasta 4x4 con constelaciones desde QPSK hasta 256-QAM.

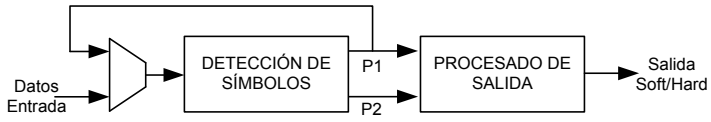


Figura 4.1: Diagrama general del detector ESPA

Este esquema no contempla la fase de preprocesado que se muestra en la figura 2.3, en la cual se realizarían las tareas de descomposición QR, para posteriormente realizar la inversión de la matriz de canal. Es importante resaltar que para el desarrollo del algoritmo SPA no es necesario la triangularización del problema. Si se realiza la descomposición QR en la fase de preprocesado es como paso previo a la inversión de la matriz de canal, la cual sí es necesaria para el desarrollo del algoritmo.

El bus, "Datos Entrada", canaliza los datos de entrada al decodificador, que son: la matriz de canal $\mathbf{H}$ estimada por el receptor, la matriz estimadora 'zero-forcing' $\mathbf{G} = \mathbf{H}^{-1}$ y el flujo de datos recibidos $\mathbf{y}_i$, objeto de decodificación. En la práctica la matriz $\mathbf{H}$, y por lo tanto su inversa $\mathbf{G}$, solo cambiarán cuando haya una variación destacable del canal, mientras que el

flujo de datos recibidos se considera a todos los efectos constante e igual a la tasa máxima alcanzada por el decodificador.

La decodificación de los datos, $\mathbf{y}_i$, se realiza en tres fases: en primer lugar el bloque de detección de símbolos de la figura 4.1 realiza la generación de vectores candidato. El número de vectores candidato es programable entre 1 y 8. En segundo lugar se realiza el cálculo de la métrica asociada a cada uno de estos vectores para finalmente, de entre todos ellos, obtener la solución *hard output*, seleccionando el de menor métrica, o *soft output*, calculando el LLR. Las dos últimas fases son realizadas por el bloque de procesado de salida de la figura 4.1 a través del algoritmo 3.7, descrito en la sección 3.4.

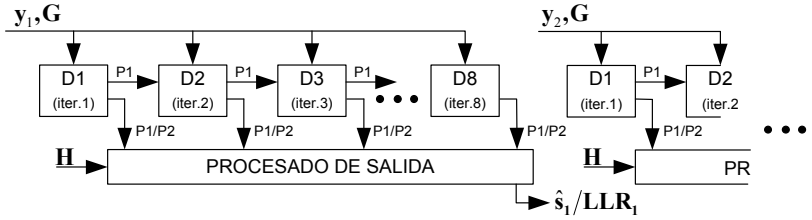


Figura 4.2: Descripción del proceso secuencial de detección ESPA

La figura 4.2 muestra el proceso secuencial de detección del detector Extended-SPA programado con 8 iteraciones. Su funcionamiento se describe con detalle a continuación:

- El bloque de detección (D) se ejecuta secuencialmente hasta 8 veces para obtener en cada iteración un vector candidato que es entregado símbolo a símbolo al bloque de procesado de salida.
- Cada una de las iteraciones (Dx) realizadas, salvo la primera (D1), recibe como datos de entrada $\mathbf{y}_i$, $\mathbf{G}$ y el *path*, pareja compuesta por el índice y el símbolo (i_k, γ_{i_k}). Este *path* corresponde al primer símbolo detectado de cada iteración y es determinado por la iteración anterior. Esto significa que, para un sistema MIMO 4x4, la primera iteración obtiene 8 símbolos reales mientras que el resto sólo 7, ya que el primer símbolo detectado, (i_k, γ_{i_k}), lo reciben de la iteración anterior.
- La comunicación entre una iteración y la siguiente se realiza a través del bus P1, según se observa en las figuras 4.1 y 4.2, mientras que la comunicación con el bloque de procesado de salida es a través de los buses P1 y P2. Esto es necesario ya que al final de cada iteración el bloque de detección de símbolos (D) obtiene simultáneamente dos datos, *"paths"*: el último de la iteración actual y el primero de la siguiente, y ambos tienen que ser transferidos al bloque de procesado de salida.

- El bloque de procesado de salida computa las métricas, el *hard output* y el *soft output* de los vectores candidato de cada iteración de forma acumulativa, símbolo a símbolo. De esta manera cuando procesa el último símbolo de la última iteración devuelve las salidas *hard output* y/o *soft output*, sin necesidad de realizar operaciones extra que penalicen la latencia del sistema.

Con todo lo visto cabe hacer las siguientes consideraciones sobre el diseño de la arquitectura:

1. La tasa de detección T (*Throughput*) alcanzada por cualquier detector es directamente proporcional la número de bits procesados e inversamente proporcional al número de ciclos de reloj necesarios para procesarlos. Para un sistema MIMO $M_t \times N_r\, q-QAM$ la tasa de detección alcanzada por el detector ESPA (figura 4.1) puede resumirse con la siguiente expresión:

$$T(Mbps) = \frac{M_t \cdot log_2(q)}{(ND_{Iter} \cdot ND_{Ciclos}) + NP_{Ciclos}} f_{clock}(MHz), \quad (4.1)$$

   donde ND_{Iter} y ND_{Ciclos} corresponden, respectivamente, al número de iteraciones y al número de ciclos de reloj necesarios para computar el bloque de detección de símbolos y NP_{Ciclos} es el número de ciclos de reloj del bloque de procesado de salida.

2. El detector ESPA a implementar, descrito por el algoritmo 3.5, requiere la multiplicación de vectores de dimensión 8 como máximo, ya que se ha establecido como especificación del diseño que el detector pueda procesar sistemas MIMO 4x4, lo que implica vectores de 8 componentes reales. Existen dos alternativas para realizar esta multiplicación vectorial:

   a) Utilizar 8 multiplicadores operando en paralelo que invierten 1 ciclo de reloj en realizar la operación.

   b) Utilizar un único multiplicador que invierte 8 ciclos de reloj para completar el cálculo.

   Para reducir el número de multiplicadores, y por lo tanto, los recursos *hardware* utilizados, se ha optado por la segunda opción.

3. El hecho de que la unidad de detección de símbolos sea iterativa (los datos de entrada de una iteración son el resultado de la anterior) impide su paralelización como herramienta para aumentar el *throughput*. Por lo tanto, esta unidad sólo puede ser implementada operando en bucle.

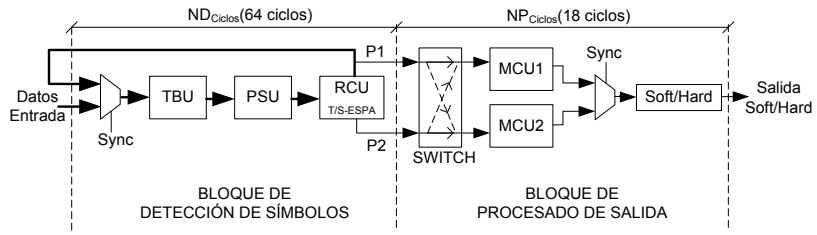


Figura 4.3: Diagrama de bloques detallado del detector ESPA

4. Se establece como mecanismo para incrementar el *throughput* del detector, operando en bucle para minimizar el área, la utilización de la técnica *pipeline-interleaving*, que consiste en procesar N_{SAMP} datos $\mathbf{y}_i$ de entrada en cadena. Este factor afecta directamente al numerador de la ecuación 4.1, quedando multiplicado el *throughput* por N_{SAMP}.

A la vista de estas consideraciones procede en este punto calcular el número de de datos N_{SAMP} que es posible procesar en el *pipeline*. Para ello, el punto de partida es conocer la profundidad del *pipeline* (ND_{Ciclos}), es decir el número de ciclos de reloj necesario para la computación del bloque de detección de símbolos, que una vez implementado se ha determinado que es 64 ciclos. Dado que los multiplicadores vectoriales consumen 8 ciclos de reloj, se concluye que el *pipeline* puede procesar $ND_{Ciclos}/8 = 8$ datos en *interleving*. Esto significa que la tasa de detección de la ecuación 4.1 quedará multiplicada por 8.

La figura 4.3 muestra con mayor nivel de detalle el diagrama de bloques del detector Extended-SPA. Los dos grandes bloques, de detección de símbolos y de procesado de salida, han sido subdivididos en unidades de procesamiento más pequeñas, cuya descripción, en profundidad, va a ser abordada en las secciones posteriores.

4.2. Bloque de detección de símbolos

La misión del bloque de detección de símbolos es obtener las componentes de los vectores candidatos, es decir, calcular el *path*: pareja compuesta por el índice y el símbolo (i_k, γ_{i_k}). Para ello se han diseñado tres unidades que están conectadas en cascada y se ejecutan en 64 ciclos de reloj en bucle, es decir, la salida de RCU está sincronizada con la entrada de TBU, según se aprecia en la figura 4.3:

Traverse Brach Unit (TBU): responsable de transformar los datos $\mathbf{y}_i$ y $\mathbf{G}$.

Path Select Unit (PSU): responsable de calcular los datos i_k, α_{i_k}, β_{i_k}, δ_{i_k}.

Repetition Control Unit (RCU): responsable de seleccionar un *path* (i_k, γ_{i_k}) no repetido y de almacenarlo.

El lector puede observar que para un sistema MIMO $M_t \times N_r$ real, se generarán $2M_t$ componentes (8 para un sistema MIMO 4x4), es decir, el bucle se ejecutará $2M_t$ veces para obtener el primer vector candidato. El sistema es programable para obtener hasta 8 candidatos.

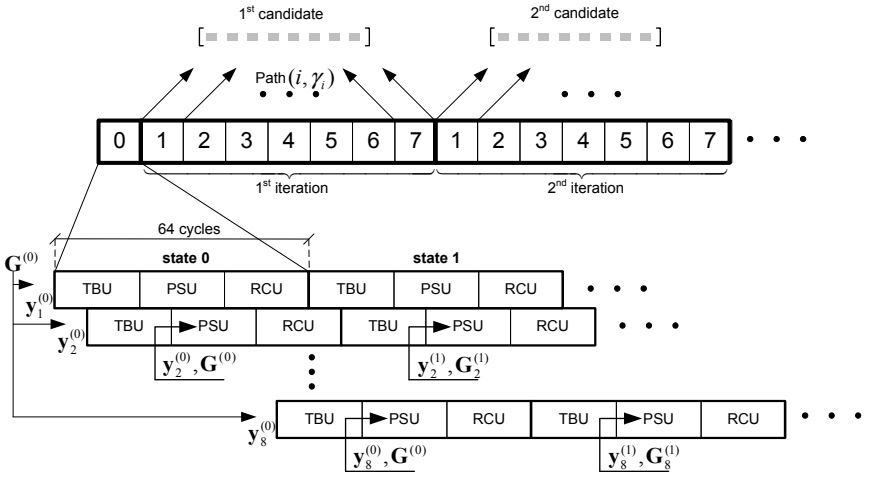


Figura 4.4: Esquema de funcionamiento del bloque de detección del símbolos para un sistema MIMO 4x4.

La figura 4.4 muestra, a través de un esquema, el detalle de funcionamiento de este bloque para un sistema MIMO 4x4. Para la correcta comprensión del esquema a continuación se define lo que denominamos estado e iteración del bloque detector de símbolos.

Estado: es la ejecución completa del bucle, es decir, de las tres unidades conectadas en cascada: TBU, PSU y RCU. Las componentes vectoriales *"paths"* se obtienen al final de cada estado.

Iteración: es la ejecución sucesiva de estados hasta obtener un vector candidato. El lector puede observar, en la parte superior de la figura, cómo se forman los vectores candidatos agrupando las componentes vectoriales. Esta última tarea se realiza en el bloque de procesado.

En cada estado de detección se calcula un *path* salvo en el último estado de cada iteración que se calcula el último *path* de esa iteración y se obtiene el

primero de la siguiente iteración, según indica el algoritmo Extended-SPA. Esto implica que la primera iteración necesite de un estado 0 que calcule el primer *path*, mientras que el resto de iteraciones no lo necesitan.

El modo de operación del bloque es *pipeline-interleaving*, con una capacidad de proceso de 8 datos, es decir, un dato cada 8 ciclos de reloj. La operación del *pipeline* se muestra en la parte inferior de la figura 4.4. Las flechas indican cómo se van transformando los datos a lo largo del proceso.

En el estado 0 los datos entran en el *pipeline*. Cada 8 ciclos de reloj entran $\mathbf{y}_i^{(0)}$ y $\mathbf{G}^{(\mathbf{0})}$. En el estado 0 la unidad TBU no transforma estos datos. Simplemente los conecta, a través de una línea de retardo seguida de un multiplexor, con la unidad PSU. En el resto de estados estos datos sí se transforman.

En el estado 1 del resto de iteraciones los datos de entrada vuelven a ser los originales: $\mathbf{y}_i^{(0)}$ y $\mathbf{G}^{(\mathbf{0})}$. En este caso la unidad TBU sí los transforma haciendo uso de la información (*paths*), calculada por la unidad PSU, que arroja la anterior iteración.

4.2.1. Unidad "Traverse Branch Unit" (TBU)

La unidad TBU forma parte del bloque de detección de símbolos y es la responsable de transformar los datos $\mathbf{y}_i$ y $\mathbf{G}$: en el estado (k) ($k \in \{1, 2, \ldots, 2M_t - 1\}$), la unidad TBU transforma los datos $\mathbf{G}^{(k-1)}$ e $\mathbf{y}^{(k-1)}$, utilizando el *path* $(i_{k-1},\ \gamma_{i_{k-1}})$ calculado por la unidad PSU en el estado anterior, en $\mathbf{G}^{(k)}$ e $\mathbf{y}^{(k)}$, según se aprecia en la figura 4.5.

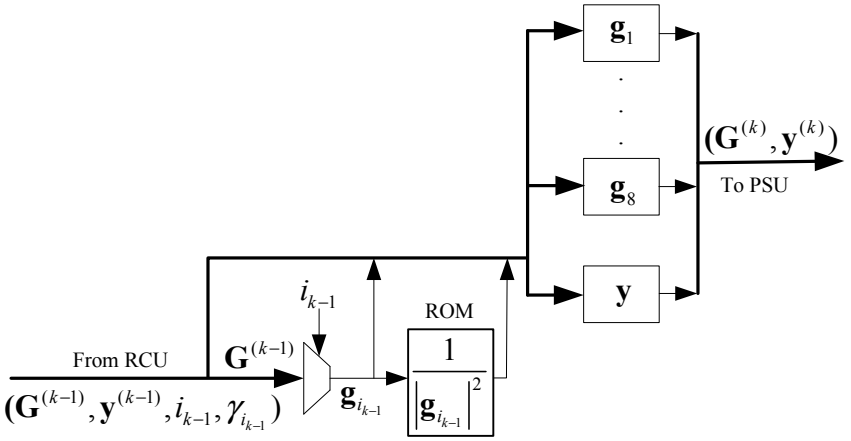


Figura 4.5: Diagrama de bloques de la unidad TBU

Esta unidad se ejecuta en 18 ciclos de reloj. Los bloques $\mathbf{g}_i$ e $\mathbf{y}$ operan

en paralelo y realizan los cálculos de las sentencias a y b de la función TraverseBranch, descrita en la sección 3.2, es decir, la cancelación del símbolo detectado sobre la proyección de $\mathbf{y}$ y la proyección de la matriz $\mathbf{G}$. Los datos de salida $\mathbf{G}^{(k)}$ e $\mathbf{y}^{(k)}$ siguen dos caminos diferentes, como puede apreciarse en la figura A.1 del anexo[1]:

1. Por una parte, son dirigidos a una línea de retardo de 46 ciclos de reloj para, a su salida, volver a ser procesados por la misma unidad TBU cerrándose así el bucle de la unidad de detección de símbolos. El lector puede observar cómo los 64 ciclos del bucle se completan con la suma de los 18 de la unidad TBU y de los 46 de la línea de retardo.

2. Por otra parte, son dirigidos a la siguiente unidad (PSU) para realizar el cálculo de los *paths* correspondientes a cada nivel.

La implementación de los bloques $\mathbf{g}_i$ e $\mathbf{y}$ de la figura 4.5 se han realizado según la ecuación 4.2 y se muestra en el circuito de la figura 4.6.

$$\mathbf{\Gamma}^{(k)} = \mathbf{\Gamma}^{(k-1)} - \frac{\sum_{j=1}^{8} \Gamma_j^{(k-1)} \cdot g_{i_{k-1},j}^{(k-1)}}{\left|\mathbf{g}_{i_{k-1}}^{(k-1)}\right|^2} \cdot \mathbf{g}_{i_{k-1}}^{(k-1)}. \tag{4.2}$$

1. $\mathbf{\Gamma}^{(k)} = \mathbf{g_i}^{(k)}$ cuando $\mathbf{\Gamma}^{(k-1)} = \mathbf{g_i}^{(k-1)}$

2. $\mathbf{\Gamma}^{(k)} = \mathbf{y}^{(k)}$ cuando $\mathbf{\Gamma}^{(k-1)} = \mathbf{y}^{(k-1)} - \mathbf{h}_{i_{k-1}}\gamma_{i_{k-1}}$

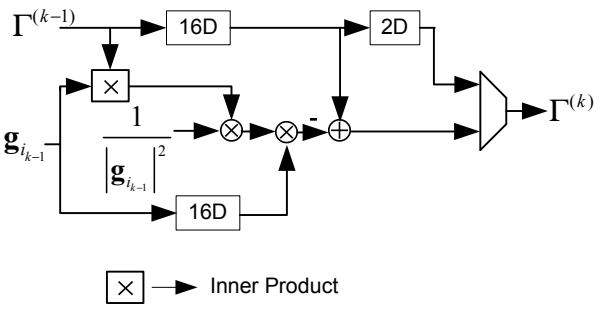


Figura 4.6: Detalle de computación de $\mathbf{g}_i$ e $\mathbf{y}$

[1] En la nomenclatura del anexo aparece, como inversa, la matriz $\mathbf{L}$ en lugar de $\mathbf{G}$. Esto es debido a que la inversión de la matriz $\mathbf{H}$ se ha realizado a través de la descomposición QR.

El producto escalar representado en la figura con el símbolo X se ha implementado con un único multiplicador-acumulador. Por ello, requiere de 8 ciclos de reloj para completar su operación.

En las figuras 4.5 y 4.6 se muestra que es común a todos los bloques $\mathbf{g}_i$ e $\mathbf{y}$ el uso de la división por $|\mathbf{g}_{i_{k-1}}|^2$. Esta división se ha resuelto mediante la tabulación de su inversa (ROM), previa selección mediante un multiplexor de la fila i_{k-1} de la matriz $\mathbf{G}^{(k-1)}$.

La línea de retardo de la parte superior de la figura 4.6, como ya se ha comentado, es necesaria debido a que en el estado (0) los datos de entrada y salida son los mismos $\mathbf{G}^{(0)}$ e $\mathbf{y}^{(0)}$, ya que en este estado no se ha calculado todavía ningún *path*. El multiplexor de la salida selecciona los datos que provienen de la línea de retardo superior en el estado (0). En cualquier otro estado el multiplexor seleccionará los datos de la ruta inferior.

4.2.2. Unidad "Path Select Unit" (PSU)

La unidad PSU implementa los cálculos que realiza la función PathSelect, descrita en la sección 3.2. Se ejecuta en 42 ciclos de reloj, siendo sus datos de entrada $\mathbf{G}^{(k)}$ e $\mathbf{y}^{(k)}$ y los de salida i_k, α_{i_k}, β_{i_k} y δ_{i_k}. A partir de estos datos la unidad RCU (Repetition Control Unit) determinará el *path* $(i_k,\ \gamma_{i_k})$ del estado (k). Su esquema de funcionamiento se muestra en la figura 4.7.

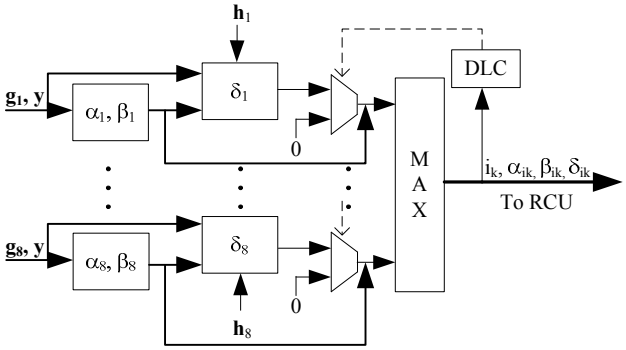


Figura 4.7: Diagrama de bloques de la unidad PSU

Esta unidad se divide básicamente en 4 módulos:

1. Cálculo de α y β
2. Cálculo de δ
3. Comparador (MAX)

4. Detected Level Control (DLC) y multiplexor de entrada al comparador

Los bloques α, β, δ y los multiplexores de entrada al comparador operan en paralelo, estando sincronizados a la entrada del comparador MAX.

El módulo para el cálculo de α y β corresponde a las sentencias *a*, *b* y *c* de la función PathSelect (sección 3.2). Sus entradas son $\mathbf{g}_i$ e $\mathbf{y}$, y las salidas α y β. La operación de los bloques se realiza en paralelo para cada fila de la matriz $\mathbf{G}$. Este bloque se computa en 23 ciclos de reloj. Su esquema de funcionamiento se muestra en la figura 4.8.

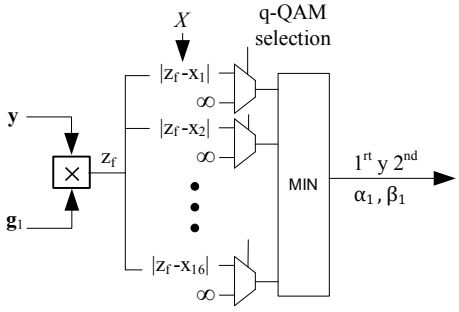


Figura 4.8: Detalle del cálculo de α y β

La salida β se utiliza para computar el *base weight* δ y paralelamente α y β son registrados y sincronizados con la entrada del comparador MAX (Fig.4.7). La arquitectura de este bloque se ha diseñado para que el detector opere con cualquier esquema de modulación QAM, desde QPSK a 256-QAM: los multiplexores situados a la entrada del comparador anulan los símbolos correspondientes cuando la selección es inferior a 256-QAM.

El producto de $\mathbf{y}$ por $\mathbf{g}_i$ se realiza mediante un único multiplicador-acumulador que requiere de 8 ciclos de reloj para realizar la operación. El resultado, z_f, es el *zero forcing* y se ha diseñado un comparador (MIN) de doble árbol para obtener los símbolos α y β, más cercanos a z_f.

El módulo para el cálculo de δ implementa la sentencia *d* de la función PathSelect que se ha realizado según las ecuaciones 3.4 y 3.5. Su esquema se muestra en la figura 4.9.

Este bloque se computa en 15 ciclos de reloj. Las entradas son $\mathbf{g}_i$, $\mathbf{y}$, $\mathbf{h}_i$, y β (calculado en el módulo anterior). Las dos multiplicaciones vectoriales se realizan a través de multiplicadores-acumuladores únicos, es decir, requieren de 8 ciclos de reloj para realizar la operación,

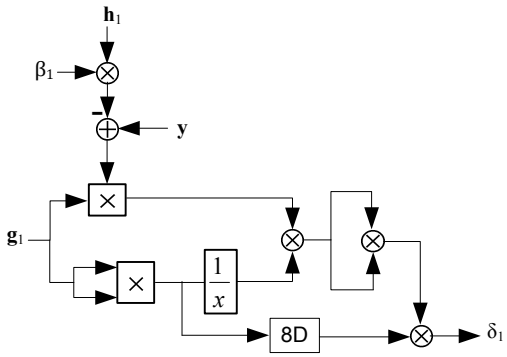


Figura 4.9: Detalle del cálculo de δ

y la implementación de la división por $|\mathbf{g}_{ik}|^2$ se ha resuelto, al igual que en la unidad TBU, mediante la tabulación de su inversa (ROM).

La implementación del comparador MAX corresponde a la sentencia e de la función PathSelect. Las entradas del comparador son vectores formados por i, α, β y δ, siendo i una constante $(1, 2, \ldots, 8)$ para cada línea de computación en paralelo. El comparador ha sido implementado en árbol y obtiene en 4 ciclos el vector correspondiente al máximo δ.

La implementación del bloque DLC y del multiplexor de entrada al comparador MAX, corresponde a la sentencia f de la función PathSelect.

Los multiplexores de entrada a la función MAX utilizan como señales de control las provenientes del bloque DLC. La misión de estos multiplexores es anular los niveles ya detectados en estados anteriores, poniendo un cero a la entrada del comparador MAX.

El bloque DLC es un registro de desplazamiento de 64 bits (8 datos de 8 símbolos cada dato). El registro de desplazamiento se inicializa con '1'. Al detectarse el nivel i del dato x se escribe un '0' en el bit correspondiente, de esta manera en la siguiente detección el multiplexor correspondiente recibirá la orden de anular el nivel ya detectado. En el estado 1 de cada iteración se resetea el registro de desplazamiento y se marca a '0' el bit correspondiente al primer nivel detectado en dicha iteración.

4.2.3. Unidad "Repetition Control Unit" (RCU)

La misión de esta unidad dentro del bloque de detección de símbolos es la de seleccionar los *path* P1 y P2 a partir de los datos de entrada provenientes del comparador de la unidad PSU (i_k, α_{i_k}, β_{i_k} y δ_{i_k}) y de los datos previamente almacenados. Como el objetivo general del algoritmo es obtener vectores candidatos no repetidos, la misión de esta unidad es evitar que el primer *path* (estado 1, Fig.4.4) de cada iteración sea distinto al de iteraciones anteriores. De esta manera se reducen las repeticiones de candidatos.

Se ha realizado la implementación de dos mecanismos de control de repeticiones correspondientes a los algoritmos Table-ESPA, desarrollado en la sección 3.3.1, y Simplified-ESPA, expuesto en la sección 3.3.2.

La unidad arroja como salida dos *paths* (P1 y P2), que son las componentes del vector solución de la iteración en curso. El bloque de procesado (Fig.4.3) gestiona la llegada de estas dos soluciones simultáneamente. Esto se produce porque en el estado de detección 7 (Fig.4.4) se generan simultáneamente el último *path* (P2) de la iteración en curso y el primero (P1) de la siguiente iteración.

Es común a los dos mecanismos de control implementados que en el estado 1 de cada iteración el *path* de salida P1 sea (i, β_i), mientras que en el resto de estados P1 y P2 son coincidentes e iguales a (i_k, α_{i_k}). Además, P1 se sincroniza no sólo como entrada del bloque de procesado sino como entrada de la unidad TBU junto con los datos $\mathbf{G}^{(k)}$, $\mathbf{y}^{(k)}$, cerrándose así el lazo del bloque de detección de símbolos.

Control de repeticiones Table-ESPA. La figura 4.10 muestra el funcionamiento de la unidad de control de repeticiones RCU en este caso. Esta unidad se ejecuta en 3 ciclos de reloj.

El control de los *paths* primarios atravesados en iteraciones anteriores se realiza mediante una tabla. La tabla almacena hasta 8 *paths* (uno por cada iteración programada) y se inicializa en el estado 0 con el primer *path* calculado en la iteración 0. La unidad RCU evalúa en cada estado de detección (desde el 1 hasta el 7) los datos de entrada para así realizar la previsión de cúal será el mejor *path* para iniciar la siguiente iteración.

En cada estado de detección el *path* (i_k, β_{i_k}) se escribe en la tabla (Fig.4.10-iii) en la posición correspondiente a la siguiente iteración. Por ejemplo, en la iteración 0 se escribirá en la posición 1 de la tabla. La posición 0 de la tabla ha sido previamente inicializada con el *path* 0.

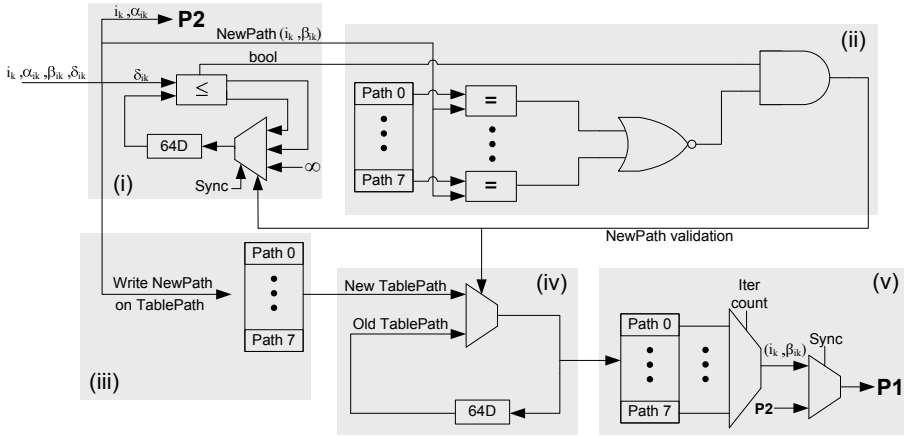


Figura 4.10: Diagrama de bloques de la unidad RCU

Como el bloque de detección de símbolos es un *pipeline* donde se están procesando simultáneamente 8 datos, es necesario mantener una tabla para cada uno de esos 8 datos. Estas tablas se mantienen en un registro de desplazamiento, y se sincronizan con la unidad RCU cada 64 ciclos de reloj (Fig.4.10-iv). Esta sincronización consiste en mantener la tabla antigua o sustituirla por la nueva si se cumplen estas dos condiciones lógicas:

1. La distancia δ_{i_k} (*base weight*) es menor que la misma del *path* evaluado en el estado anterior (Fig.4.10-i).
2. El *path* evaluado es distinto a los anteriores almacenados en la tabla (Fig.4.10-ii).

Este mecanismo de control se muestra eficiente para las 8 primeras iteraciones. De ahí que la arquitectura *hardware* del detector se haya limitado para almacenar datos hasta un máximo de 8 iteraciones.

Control de repeticiones Simplified-ESPA . La figura 4.11 muestra el funcionamiento de la unidad de control de repeticiones RCU en este caso. Esta unidad se ejecuta en 3 ciclos de reloj.

Su funcionamiento se basa en un registro de desplazamiento que almacena, para cada uno de los 8 datos procesados por el pipeline, el *path* $(i,\ \beta_i)$ cuyo δ_i es menor. El registro de desplazamiento inicializa con ∞ los *"paths"* tomados en el estado 1, al final de cada iteración tras el estado 7. De esta manera entre una iteración y la siguiente no se repite el primer *path* tomado.

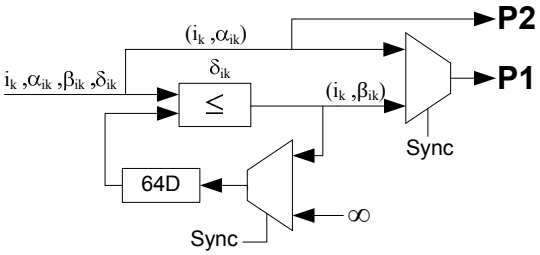


Figura 4.11: Diagrama de bloques de la unidad RCU

Este mecanismo selecciona el *path* con que arrancar cada iteración en el estado 1, atendiendo exclusivamente al criterio de distancia, y sólo garantiza la no repetición del primer *path* entre iteraciones consecutivas. Por lo tanto, su utilización es útil para una configuración de como máximo 5 iteraciones.

4.3. Bloque de procesado de salida

Como ya se ha comentado en los apartados anteriores, este detector obtiene como salidas el *hard output* o el *soft output*, según la implementación realizada, a través del algoritmo 3.7 expuesto en la sección 3.4.

La misión del bloque de procesado (Fig.4.3) es agrupar las componentes vectoriales (*paths*) calculados en el bloque de detección de símbolos para formar los vectores candidatos, calcular la métrica asociada a cada vector, y devolver el vector cuya métrica sea menor en el caso de la implementación *hard output* u obtener el LLR para el *soft output*.

Por lo tanto, el bloque de procesado se compone de tres unidades básicas:

Unidad de doble computación de métrica (MCU1, MCU2): responsable del cálculo de las métricas $d(\mathbf{s}_i)$ asociadas a cada uno de los vectores candidato obtenidos en las diferentes iteraciones.

Unidad *hard output* (Hard): responsable de devolver, de entre todos los vectores candidato $\mathbf{s}_i$, aquel cuya métrica sea menor.

Unidad *soft output* (Soft): responsable de calcular el LLR a partir de los vectores candidato y de sus métricas $d(\mathbf{s}_i)$ asociadas.

4.3.1. Unidad de doble computación de métrica

Esta unidad genera para cada uno de los 8 datos procesados por el *pipeline* un vector candidato por cada iteración programada. Estos vectores

se entregan a la unidad *soft output* o a la unidad *hard output* según sea la implementación realizada. Para ello, la unidad de doble computación de métrica tiene como entradas los *"paths"* (P1 y P2) generados por el bloque de detección de símbolos y como salidas en la iteración j los vectores candidatos $\mathbf{s}_j$, con su métrica asociada $d(\mathbf{s}_j)$. La figura 4.12 muestra el esquema de funcionamiento de esta unidad, que se ejecuta en 13 ciclos de reloj.

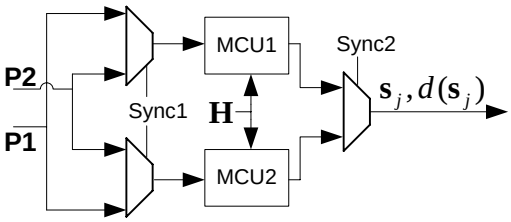


Figura 4.12: Diagrama de bloques de la unidad DMCU

Se han utilizado dos unidades de computación de métrica (MCU) en paralelo ya que al final de cada iteración se producen simultáneamente dos datos: el último *path* de la iteración en curso y el primero de la siguiente. Los multiplexores de entrada a la unidad distribuyen los datos de entrada de la siguiente manera:

1. En las iteraciones impares trabaja la MCU1 y en las pares trabaja la MCU2.

2. En el estado 1 de cada iteración trabajan las dos simultáneamente.

3. El primer *path* de entrada a cada MCU en cada iteración es P2, el resto de *paths* son siempre P1, salvo en la primera iteración que son todos P1.

La figura 4.13 muestra con detalle el esquema de implementación de la unidad MCU, que realiza el cálculo de la distancia según la ecuación 2.8 ($d(\mathbf{s}) = \|\mathbf{y} - \mathbf{Hs}\|^2$). La distancia total se calcula acumulando la contribución de las distancias parciales de cada componente vectorial, y sincronizándolas a través de un registro de desplazamiento de 64 ciclos.

El módulo conformador de vectores (Vector Formation) ordena las componentes vectoriales de cada vector candidato y realiza la conversión serie/paralelo.

Cada 512 (64x8) ciclos de reloj la unidad arroja 8 vectores candidatos con sus 8 distancias, correspondientes a los 8 datos que se procesan en modo *pipeline*.

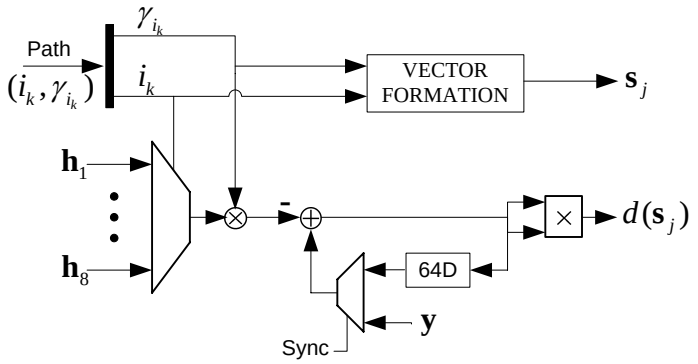


Figura 4.13: Detalle del cálculo de la distancia

4.3.2. Unidad *hard output*

La unidad *hard output* obtiene como salida, de entre todos los candidatos $\mathbf{s}_j$, aquel que tenga la métrica $d(\mathbf{s}_j)$ menor. La figura 4.14 muestra con detalle el circuito con el que se obtiene esta solución.

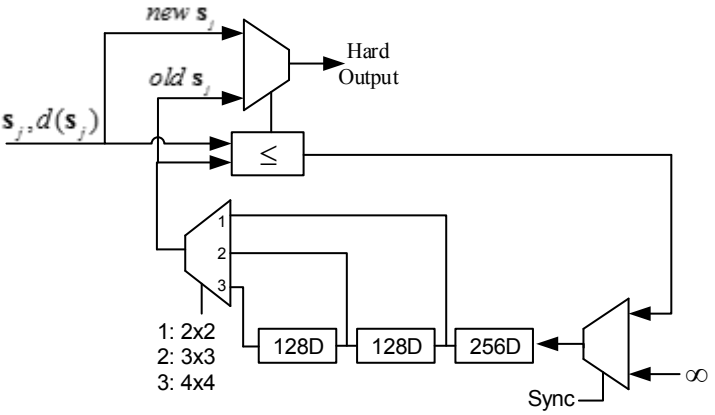


Figura 4.14: Detalle de la obtención de la solución *hard output*

Un comparador se encarga de seleccionar la menor métrica. La comparación se realiza entre el nuevo candidato y el anterior almacenado. El comparador genera dos salidas:

- La menor de las dos métricas, que es direccionada a una línea de retardo a través del multiplexor de la parte inferior de la figura, cuya función es la inicialización de las métricas.

- El bit de control del multiplexor de la parte superior de la figura, responsable de seleccionar la salida *hard output* entre los dos candidatos.

El resultado de la comparación entra en una línea de retardo que ofrece tres posibilidades de sincronismo con el siguiente candidato. Esto es debido a que la arquitectura ha sido diseñada para procesar sistemas MIMO con configuraciones de antenas 2x2, 3x3 y 4x4, cuyos vectores contienen 4, 6 y 8 símbolos reales, respectivamente. Para un sistema MIMO 4x4 la siguiente comparación se realizará a los 512 ciclos de reloj, es decir, tras la obtención de 8 símbolos que se computan tras 64 ciclos de reloj cada uno, que es la profundidad del *pipeline* del bloque de detección de símbolos. Esta unidad se ejecuta en 5 ciclos de reloj.

4.3.3. Unidad *soft output*

La detección *soft output* tiene como objetivo encontrar la información APP de cada bit utilizando el valor *"Log-Likelihood Ratio"* (LLR), cuya simplificación apta para la implementación hardware está representada por las ecuaciones (Ec.2.5 y 2.9).

La figura 4.15 muestra el diagrama general de bloques de la unidad, que tiene como misión obtener la salida *soft output* a partir de los vectores candidatos $\mathbf{s}_j$ y su métrica asociada $d(\mathbf{s}_j)$. Esta unidad ha sido desarrollada siguiendo el método expuesto en la sección 3.4.

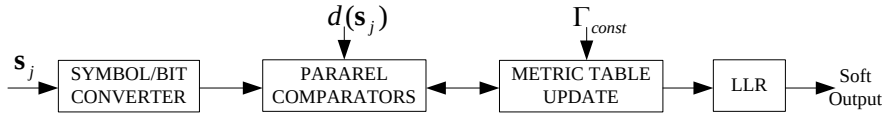


Figura 4.15: Diagrama de bloques de la unidad *soft output*

1. El módulo 'Symbol/bit Converter' recibe los vectores $\mathbf{s}_j$ de la unidad de doble computación y realiza la asignación de bits de los símbolos QAM, según indica la figura 4.16.

   Las partes real e imaginaria del símbolo direccionan una memoria donde se almacena los bits correspondientes a cada símbolo. En función de la modulación programada en el detector (QPSK, 16QAM, 64QAM o 256QAM), se obtendrán 2, 4, 6 u 8 bits respectivamente.

   Los valores L de cada bit se calculan según la siguiente expresión:

$$L\left(b_{l,k}|\mathbf{y_c}\right) \approx (\Lambda_{l,k}^{-1} - \Lambda_{l,k}^{+1}),$$

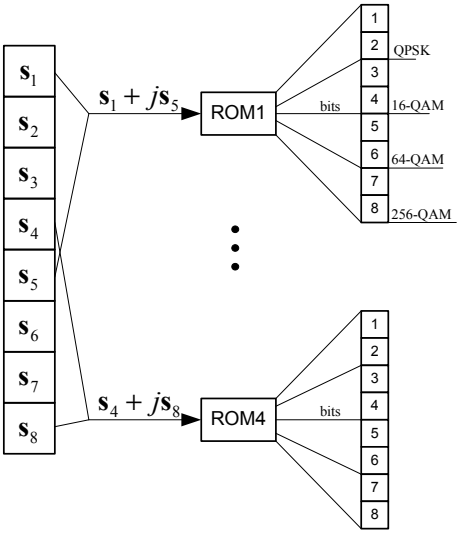


Figura 4.16: Detalle del circuito generador de bits

donde $b_{l,k}$ representa el k-ésimo bit del l-ésimo símbolo y $\Lambda = \dfrac{d(\mathbf{s}_j)}{N_0}$. Para lo cual se utilizará la tabla de la figura 4.17.

	$b_{1,1}$	$b_{1,2}$		$b_{l,k}$		$b_{4,8}$
+1	$\Lambda_{1,1}^{+1}$	$\Lambda_{1,2}^{+1}$	• • •	$\Lambda_{l,k}^{+1}$	• • •	$\Lambda_{4,8}^{+1}$
−1	$\Lambda_{1,1}^{-1}$	$\Lambda_{1,2}^{-1}$		$\Lambda_{l,k}^{-1}$		$\Lambda_{4,8}^{-1}$

Figura 4.17: Detalle de la tabla de métricas.

Dado el vector de símbolos $\mathbf{s}_j$ de la iteración j y su métrica asociada $d(\mathbf{s}_j)$, el valor Λ será escrito en las posiciones de la tabla correspondientes a los bits asociados a los símbolos del vector. Se escribirá el valor Λ en su posición correspondiente cuando su valor sea mejor que el anterior almacenado en la tabla (mejor métrica). Para controlar si una métrica es mejor que la anterior se utilizará una batería de comparadores en paralelo.

Una vez procesados todos los vectores candidatos puede ocurrir que alguna posición de la tabla se haya quedado sin escribir. En este caso las posiciones serán escritas con el valor constante Λ_{const}, según se expone en la sección 3.4.

Esta unidad, al igual que la unidad *hard output*, se ejecuta en 5 ciclos de reloj, ya que el producto $\Lambda = \frac{d(\mathbf{s}_j)}{N_0}$ se puede evitar efectuándolo en la fase de preprocesado a los datos de entrada $\mathbf{y_i}$ y a la matriz de canal $\mathbf{H}$.

4.4. Resultados de implementación

Para la implementación *hardware* de los decodificadores se ha utilizado el lenguaje de descripción *hardware* VHDL. Los modelos *hardware* primero se han verificado funcionalmente a nivel de código, comparándolos con los modelos de precisión finita previamente desarrollados. Esta verificación se ha realizado con la herramienta Modelsim de Mentor Graphics.

Una vez validado el modelo *hardware*, éste se ha implementado en un ASIC y en un dispositivo FPGA. Las implementaciones en un ASIC se han realizado usando la librería de celdas estándar Faraday de 90nm con 8 capas metálicas [22]. Para la síntesis se ha empleado la herramienta de Cadence RTL *compiler* y para el emplazado y rutado la herramienta SOC *encounter* de Cadence, la cual integra la herramienta para el análisis de tiempos. Las implementaciones en FPGA se han realizado con la herramienta ISE de Xilinx, las cuales integran sus respectivas herramientas de análisis de tiempo. Los modelos post emplazado y rutado generados por las diferentes herramientas se han verificado funcionalmente siguiendo el mismo flujo que para la verificación a nivel de código.

Se han realizado tanto para FPGA como para ASIC 4 implementaciones de este detector en función del tipo de salida obtenida *hard output* o *soft output* y del mecanismo de control de repeticiones utilizado:

1. *Hard output* / Table-ESPA (H-TESPA)

2. *Hard output* / Simplified-ESPA (H-SESPA)

3. *Soft output* / Table-ESPA (S-TESPA)

4. *Soft output* / Simplified-ESPA (S-SESPA)

de esta manera el bus de salida arrojará un vector con métrica mínima de entre todos los candidatos en el caso de las implementaciones *hard output*, o el LLR, es decir, los bits con las APP's asociadas, en el caso de las implementaciones *soft output*.

4.4.1. Pérdidas de implementación

La precisión de los datos ha sido establecida para minimizar las pérdidas de implementación, además de para compatibilizar las dos implementaciones realizadas (FPGA y ASIC). Todos los datos se han cuantificado con 18 bits (tamaño de los multiplicadores embebidos en las FPGA utilizadas), salvo dos excepciones:

1. La matriz $\mathbf{R}$, que se ha cuantificado con 13 bits para prevenir el crecimiento que experimenta al multiplicarla por los símbolos de la constelación.

2. Los datos de entrada y salida de la ROM con la que se ha tabulado la inversa de $|\mathbf{g}_{i_{k-1}}|^2$. En este caso las precisiones se han ajustado a [16,5] para los datos de entrada, bus de direcciones, y [16,11] para los datos de salida, datos almacenados en la ROM. Para maximizar la precisión en el cálculo de la inversa la ROM ha sido dividida en dos partes ROM1 y ROM2 direccionadas con los 11 bits (LSB) y los 5 bits (MSB) respectivamente. De esta manera los datos de entrada menores de 2^6 serán leídos con mayor precisión de la ROM1, mientras que el resto de datos, que no requieren tanta precisión, se leen de la ROM2.

Los bits fraccionales se han ajustado mediante simulación para conseguir las mínimas pérdidas. Finalmente, la cuantificación de la matriz $\mathbf{G}$ y del vector $\hat{\mathbf{y}}_r$ se ha ajustado a [18,10] al igual que la métrica calculada en los bloques MCU, obteniendo unas pérdidas inferiores a 0.2 dB para un BER de 10^{-3} en todas las implementaciones realizadas.

Las figuras 4.18 y 4.19 muestran el detalle de la medida de pérdidas de implementación que se ha realizado para los detectores H-TESPA configurados para sistemas MIMO 4x4 64QAM y 256QAM respectivamente.

En el anexo A se muestran los diagramas de implementación y la tabla precisiones (A.1) ajustadas para minimizar las pérdidas de prestaciones BER.

4.4.2. Implementación FPGA

La implementación en FPGA se ha realizado con la herramienta de diseño ISE, para el dispositivo XC6VLX550T de Xilinx. La tabla 4.1 resume los datos de utilización de recursos para las cuatro implementaciones realizadas, así como la frecuencia máxima de reloj alcanzada (camino crítico). La primera columna muestra el número de *slices* utilizados, la segunda los bloques de memoria y la tercera el número de unidades DSP (multiplicador/sumador).

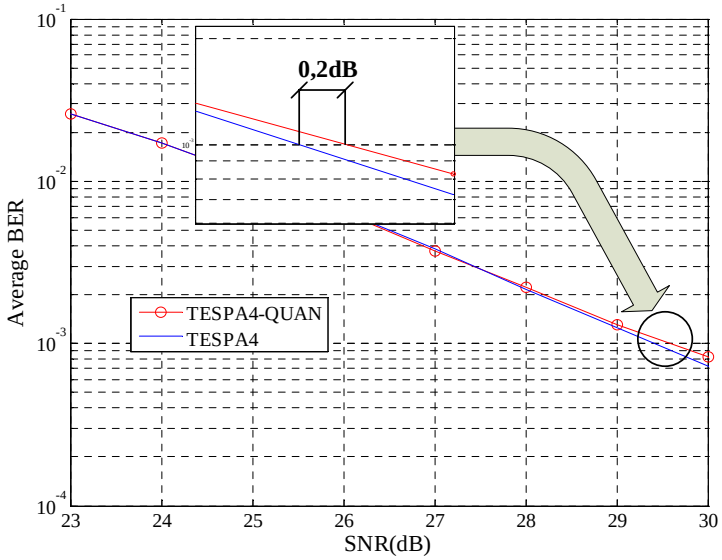


Figura 4.18: Pérdidas de implementación en prestaciones BER de detectores SPA. MIMO 4x4 64-QAM

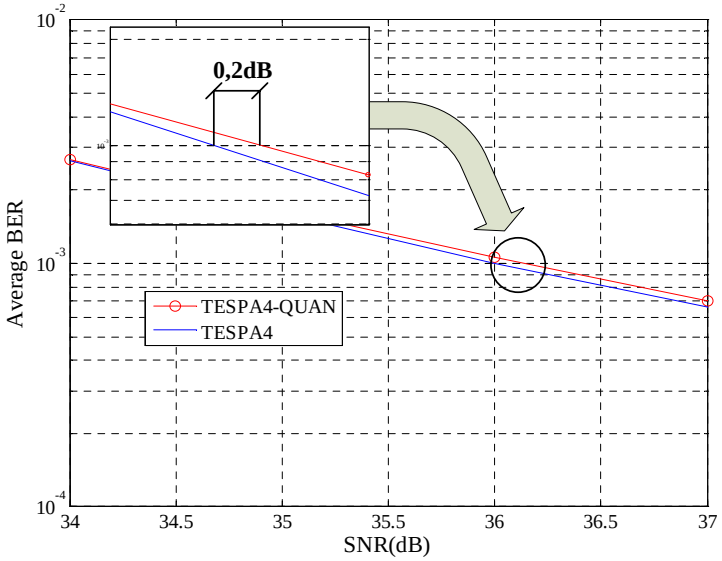


Figura 4.19: Pérdidas de implementación en prestaciones BER de detectores SPA. MIMO 4x4 256-QAM

Tabla 4.1: Utilización de recursos en FPGA

	Slices	LUT	BRAM18E1	DSP48E1	Max. Clock (MHz)
H-TESPA	6304	20371	18	81	222
H-SESPA	6368	20036	18	81	206
S-TESPA	8984	30605	18	82	197
S-SESPA	8915	30197	18	82	201

Tabla 4.2: Utilización de recursos en ASIC

	C	NC	NI	Area (mm^2)	G.E.	Max. Clock (MHz)
H-TESPA	1831341	1555480	611098	3,75	265702	806
H-SESPA	1702213	1492375	606598	3,56	248830	961
S-TESPA	2149758	1819295	753162	4,42	311634	806
S-SESPA	1896558	1583086	582362	3,83	274052	961

4.4.3. Implementación ASIC

La implementación ASIC se ha realizado con la herramienta de diseño Synopsys utilizando la libreria standard Faraday de 90nm. La tabla 4.5 resume los datos de utilización de recursos para las cuatro implementaciones realizadas, así como la frecuencia máxima de reloj alcanzada. Las tres primeras columnas son datos suministrados por la herramienta de síntesis y corresponden, respectivamente, al número de circuitos combinacionales (C), número de circuitos no combinacionales (NC) y cableado (NI). Las dos siguientes columnas, área (mm^2) y puertas equivalentes (G.E.), se calculan a partir de las anteriores, siguiendo las instrucciones suministradas por la herramienta de síntesis, a partir de las siguientes fórmulas:

$$Area(mm^2) = (C + NC + 0,6\,NI)10^{-6}$$

$$G.E. = \frac{C}{9} + \frac{NC}{25}$$

4.4.4. Tasa de detección alcanzada

La tasa de detección alcanzada por el detector dependerá del número de iteraciones programadas (ζ), entre 1 y 8, y del sistema MIMO ($N_r \times M_t$, $q - QAM$) seleccionado. El detector *hardware* ha sido diseñado para configuraciones de antenas entre 2x2 y 4x4 y para esquemas de modulación

entre QPSK ($q = 4$) y 256QAM ($q = 256$). Teniendo en cuenta que la capacidad del pipeline, según se ha establecido en (4.1), es de $N_{SAMP} = 8$, la tasa de detección T (*Throughput*) alcanzada por este detector puede ser formulada como:

$$T(Mbps) = \frac{N_{SAMP} \cdot M_t \cdot log_2(q) \cdot f_{clock}}{64(8 + 7\zeta) + 17}, \tag{4.3}$$

siendo f_{clock} (MHz) la máxima frecuencia de reloj alcanzada por el dispositivo VLSI seleccionado.

La tabla 4.3 muestra las tasas de detección alcanzadas por los detectores desarrollados en esta tesis para diferentes configuraciones. Las tasas han sido calculadas a través de la ecuación 4.3 para sistemas MIMO 4x4 ($M_t = 4$) con las modulaciones, de alta eficiencia espectral 64QAM ($q = 64$) y 256QAM ($q = 256$). Para los detectores TESPA, tanto *hard output* como *soft output*, se muestran las tasas para 1 y 8 iteraciones, mientras que para los detectores SESPA para 1 y 4. Estos límites garantizan una detección eficiente, en cuanto a prestaciones BER se refiere, según se ha establecido mediante simulaciones en el capítulo anterior.

De la tabla se desprende que las tasas máximas de detección son alcanzadas por los detectores SESPA (*hard output* y *soft output*) implementados en ASIC ($f_{clock} = 961$ MHz). En particular, con un esquema de modulación 256QAM estos detectores alcanzan un *throughput* de $465,06$ Mbps con 1 sola iteración y $131,35$ Mbps con 4.

El dispositivo FPGA (Vitex 6 de Xilinx) sobre el que se han realizado las implementaciones opera, en los cuatro diseños, con frecuencias de reloj que varían en un intervalo muy estrecho, entre 197 MHz y 222 MHz. Como consecuencia de esto, las diferencias entre las tasas de detección obtenidas se deben, fundamentalmente, a las diferentes configuraciones establecidas en cuanto a esquema de modulación y número de iteraciones se refiere. Para el caso anteriormente citado, detector SESPA *hard output* 256QAM, que opera a $f_{clock} = 206$ MHz, la tasa alcanzada es de $99,69$ Mbps con 1 iteración y $28,16$ Mbps con 4 iteraciones, que corresponde a un factor 4,66 entre las tasas obtenidas en ASIC y FPGA.

4.5. Estudio de topologías para la mejora del *throughput*

En general, los detectores MIMO sub-óptimos presentan una relación inversa entre las prestaciones BER que consiguen y su tasa de detección, es decir, alcanzan las mejores prestaciones BER a costa de reducir el *throughput*. En particular, los detectores ESPA desarrollados en esta tesis mejoran,

Tabla 4.3: Valores de *throughput* alcanzados para diferentes configuraciones en la decodificación de un sistema MIMO 4x4

Hardware	Detector	Modulación	Iteraciones	Tasa(Mbps)
FPGA	H-TESPA	64QAM	1	80,6
			8	11,6
		256QAM	1	107,4
			8	15,51
	H-SESPA	64QAM	1	74,8
			4	21,1
		256QAM	1	99,7
			4	28,2
	S-TESPA	64QAM	1	71,5
			8	10,3
		256QAM	1	95,3
			8	13,8
	S-SESPA	64QAM	1	72,9
			4	20,6
		256QAM	1	97,3
			4	27,5
ASIC	H-TESPA	64QAM	1	292,5
			8	42,2
		256QAM	1	390,1
			8	56,3
	H-SESPA	64QAM	1	348,8
			4	98,5
		256QAM	1	465,1
			4	131,3
	S-TESPA	64QAM	1	292,5
			8	42,2
		256QAM	1	390,1
			8	56,3
	S-SESPA	64QAM	1	348,8
			4	98,5
		256QAM	1	465,1
			4	131,3

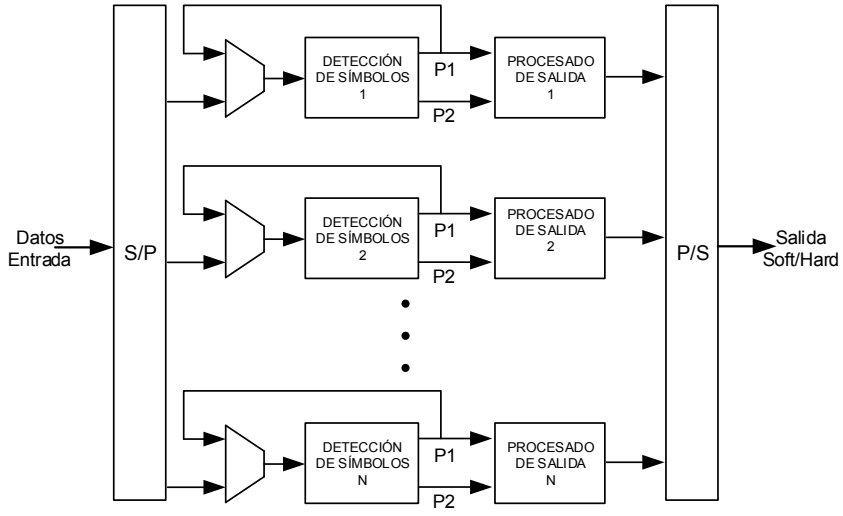


Figura 4.20: Esquema de paralelización de los detectores ESPA

en cuanto a prestaciones BER se refiere, con el número de iteraciones programadas, y esto se traduce en una disminución del *throughput*, como el lector puede apreciar en la tabla 4.3.

A estos dos parámetros de calidad hay que añadir un tercero, el área requerida por la implementación. Para unas prestaciones BER determinadas es posible incrementar la tasa añadiendo unidades del detector en paralelo, lo que penaliza, obviamente, el área. Un buen equilibrio entre prestaciones BER, *throughput* y área será determinante a la hora de seleccionar uno u otro detector.

La figura 4.20 muestra el esquema de paralelización de los detectores ESPA, donde N detectores trabajan en paralelo entre los conversores serie-paralelo y paralelo-serie respectivamente.

Si bien la paralelización es la opción más intuitiva para ganar *throughput*, la figura 4.21 muestra una topología alternativa que podría utilizarse para los detectores ESPA, donde por cada dos bloques de detección de símbolos sólo sería necesario un bloque de procesado de salida modificado. Para el buen funcionamiento de este esquema cabría hacer dos modificaciones del detector ESPA debido a la doble salida del bloque de detección de símbolos, P1 y P2.

1. Añadir un retardo de 8 ciclos en el bloque de detección de símbolos que actúa en paralelo

2. Añadir una MCU (unidad de computación de métrica) nueva en el

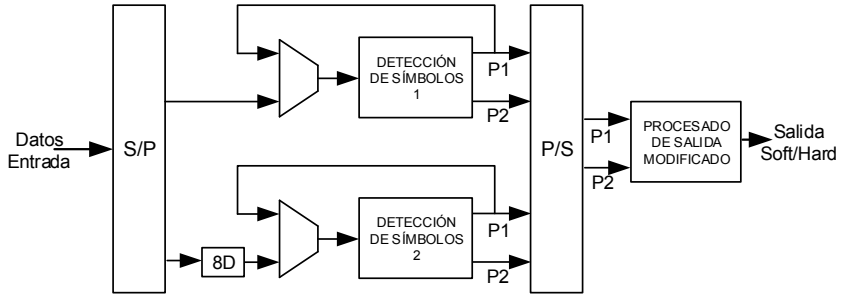


Figura 4.21: Topología alternativa de los detectores ESPA

Tabla 4.4: Posibilidades de paralelización de los detectores ASIC H-SESPA y H-TESPA para conseguir tasas superiores a 1 Gbps en MIMO 4x4 64QAM

Detector ASIC	Num. Iter.	Fact. Paralel.	Area (mm^2)	G.E. (k)	*Throughput* ($Gbps$)
H-SESPA	1	3	10,68	746	1,05
	2	6	21,35	1493	1,13
	3	8	28,47	1991	1,04
	4	11	39,14	2737	1,08
H-TESPA	5	15	56,30	3986	1,00
	6	18	67,56	4783	1,01
	7	21	78,82	5580	1,01
	8	24	90,08	6377	1,01

bloque de procesado de salida.

Con todo ello se conseguiría una mejora de área, correspondiente a un MCU, por cada dos bloques de detección de símbolos operando en paralelo. Sobre esta opción de paralelización no se ha realizado una estimación del binomio área/ *throughput* debido a que requiere modificaciones del *hardware* que no se han realizado.

La tabla 4.4 muestra la estimación del área en mm^2 y kG.E. para los detectores implementados en ASIC H-SESPA y H-TESPA paralelizados según el esquema de la figura 4.20. Para obtener los datos, los detectores, trabajando sobre MIMO 4x4 64QAM, han sido paralelizados por un factor para obtener un *throughput* superior a 1 *Gbps*. El detector H-SESPA opera hasta 4 iteraciones, mientras que el H-TESPA lo hace entre 5 y 8. Las prestacio-

nes BER alcanzadas por los detectores aumentan progresivamente con el número de iteraciones programadas, y con ellas el factor de paralelización y el área necesarios para mantener una tasa de detección superior a 1 *Gbps*.

4.6. Comparativas con otras implementaciones

Las prestaciones alcanzadas por los detectores implementados se resumen en la tabla 4.5 junto con las de los mejores detectores publicados. Todos trabajan con esquemas de modulación y número de antenas elevados y ninguno alcanza las prestaciones ML.

Los detectores propuestos en este trabajo son configurables tanto en antenas, hasta 4x4, como en esquema de modulación, desde QPSK hasta 256-QAM, este último especificado en la revisión 12 del estándar 3GPP-LTE. El detector SESPA alcanza un tasa de decodificación máxima de 465 MBps para 256-QAM con una sola iteración. Estas tasas satisfacen las tasas de pico requeridas por la próxima generación de WiMAX [2] y sistemas LTE[3].

El número de iteraciónes de los decodificadores también es programable en función de las prestaciones FER o de la tasa de decodificación requeridas por el sistema, tal y como se muestra en la figura 3.12, y varía entre 1 y 4 para el SESPA y entre 1 y 8 para el TESPA.

Para efectuar la comparación de forma coherente, la tabla se ha dividido en dos partes distinguiendo las arquitecturas especializadas de las flexibles.

- **Arquitecturas especializadas** Están diseñadas y optimizadas para conseguir las mejores prestaciones para una sola configuración de antenas y esquema de modulación. En particular, las dos seleccionadas consiguen prestaciones óptimas para sistemas MIMO 4x4 64-QAM. En [38] se presenta una arquitectura KBest que alcanza una tasa de 655 Mbps pero con unas prestaciones BER limitadas por la selección de K=10. En [20] con la misma K se consigue una tasa de 1 Gbps utilizando una modificación del algoritmo KBest, con enumeración compleja y muy sensible a la precisión de los datos, que penaliza la BER y el área. Los autores proponen como futuras lineas de trabajo la extensión de estas arquitecturas a 256-QAM con costes hardware limitados.

[2]La especificación de tasas de pico para IEEE 802.16m es [37]: (i) Tasa de datos muy baja: 16kbps, (ii) Tasa de datos baja y Multimedia baja: 144 kbps, (iii) Multimedia medio: 2Mbps, (iv) Multimedia alto: 30 Mbps, (v) Multimedia super alto: 30Mbps∼100Mbps/1Gbps.

[3]Las tasas de pico de datos en LTE son 326,4 Mbps para bajadas y 50 Mbps para subidas.

Tabla 4.5: Comparación de implementaciones de detectores MIMO

	SPECIALIZED ARCH.		**FLEXIBLE ARCHITECTURE**						
Reference	**TVLSI 2012 [38]**	**TVLSI 2013 [20]**	**TCAS-I 2009 [31]**	**TCAS-I 2009 [39]**	**JSSC 2010 [40]**	**TVLSI 2010 [21]**	**THIS WORK**	
							SESPA	**TESPA**
Antenna	4×4	4×4	2×2∼6×4	2×2∼16×16	2×2∼8×8	4×4	2×2∼4×4	
Modulation	64QAM	64QAM	(4∼64)QAM	(4∼64)QAM	(4∼64)QAM	64QAM	(4∼256)QAM	
Method	KBest	Modified KBest	OSIC/V-ML	SD/KBest	Best-First	KBest	OSIC	
Soft/Hard	Hard	Hard	Soft	Hard	Soft	Hard	Hard/Soft	
SNR-Independent	Yes	Yes	Yes	Yes	No	Yes	Yes	
Technology	130 nm	130 nm	130 nm	90 nm	130 nm	65 nm	90 nm	
Gates(kGE)	114	340	205	87	350	1760	274	312
Area (mm^2)	Core 3,24	— —	Total 5.29	Core 0.31	Core 1.77	—- —-	Total 3.83	Total 4.42
Max. clock rate	282MHz	417MHz	71MHz	256MHz	198MHz	158MHz	961MHz	806MHz
Throughput (MBps)	655 (K=10)	1000 (K=10)	114 4x4(64QAM)	1536 16x16(64-QAM)	431,8 4x4(64QAM)	732(K=5) 463(K=10) 100(K=64)	**64QAM** 98,5/348,8 **256QAM** 131,3/465	42,2/292,5 56,3/390

- **Arquitecturas flexibles**. Permiten seleccionar diferentes configuraciones de antenas, esquemas de modulación o tasa de decodificación.

  En [31] se implementa una arquitectura mixta basada en búsqueda exhaustiva para configuraciones de antenas 2x2 y OSIC para superiores. Si bien para 2x2 alcanza la diversidad ML, las prestaciones BER conseguidas para 4x4 64-QAM, basadas en VBLAST, son inferiores a las alcanzadas por los detectores desarrollados en este trabajo con una sola iteración (Fig.3.7).

  En [39] se presenta un detector basado en los algoritmos SD y KBest. Aunque los datos de área (87 kGE) y *throughput* (1536 Mbps para 16x16 64-QAM) son muy buenos, las prestaciones BER alcanzadas en estas condiciones son muy inferiores a las V-BLAST. Para mejorar esto, los autores proponen paralelizar la estructura básica del detector penalizando enormemente el área o utilizar técnicas de corrección de errores, añadiendo redundancia y penalizando así la tasa de detección.

  El detector presentado en [40] utiliza un código convolucional con un ratio de 1/2 consiguiendo una BER de 10^{-5} para una EbNo de 24.2 dB trabajando a la velocidad máxima. Sin embargo los autores admiten que si las condiciones de canal no son favorables para mantener las prestaciones de BER el *throughput* tiene que reducirse. En el caso del detector presentado en este trabajo, S-SESPA, con un decodificador LDPC y un ratio 2/3, se alcanza la BER 10^{-5} a una EbNo de 19.6 dB.

  El detector [21] presenta una arquitectura 4x4 64-QAM basada en KBest que permite la configuración del parámetro K entre 5 y 64, o lo que es lo mismo, permite seleccionar el balance entre prestaciones BER y tasa de decodificación. Sin embargo, este detector no permite la configuración del número de antenas o el esquema de modulación. El decodificador TESPA presentado en este trabajo permite también el ajuste de este balance mediante la configuración del número de iteraciones entre 1 y 8, cuya equivalencia con KBest corresponde con K=5 y K=30 respectivamente. El área de este detector supera con creces a la de cualquier otro de esta tabla debido a que que permite el ajuste de K=64, cuyas prestaciones son quasi-ML. Paralelizando en un factor 7, según el esquema de la figura 4.20, la implementación H-TESPA alcanzaría una tasa de unos 300 Mbps con un área de unos 1860 kGE consiguiendo unas prestaciones BER similares a las alcanzadas por este detector con K=30 y muy próximas a la diversidad ML.

En general los detectores SESPA y TESPA tienen un alto grado de configurabilidad alcanzando un equilibrio entre area, BER y *throughput*

muy competitivo frente a los receptores recientemente publicados, ofreciendo además las decisiones Hard y Soft en paralelo.

4.7. Conclusiones

El estudio de prestaciones realizado en el capítulo anterior sobre los detectores basados en el algoritmo SPA no sólo muestra que éstos son claramente competitivos en lo que respecta al binomio prestaciones/complejidad, sino que además, permiten ser configurados para adaptarse a diferentes sistemas MIMO en lo que respecta a antenas y esquemas de modulación.

En este capítulo se ha abordado el diseño de la arquitectura que hace viable y compatible la implementación de los mismos tanto en FPGA como ASIC. Se han desarrollado cuatro variantes del detector ESPA en función del mecanismo de control de repeticiones (3.3) y de la salida (3.4) requerida:

1. *Hard output* / Table-ESPA (H-TESPA)
2. *Hard output* / Simplified-ESPA (H-SESPA)
3. *Soft output* / Table-ESPA (S-TESPA)
4. *Soft output* / Simplified-ESPA (S-SESPA)

La arquitectura ha sido diseñada a través de los bloques de detección de símbolos y de procesado de salida utilizando la técnica de *pipeline-interleaving*. Mediante esta técnica se ha incrementado el *throughput* de los detectores en un factor 8. Se han ajustado las precisiones de los datos, consiguiendo unas pérdidas de implementación mínimas de 0,2 dB en cuanto a prestaciones BER se refiere, y se ha realizado un estudio detallado de las posibilidades de paralelización que ofrece la arquitectura. De esta manera, los detectores desarrollados pueden alcanzar tasas superiores a Gbps a costa, eso sí, de un incremento de área.

Estos detectores, además, presentan una alta flexibilidad en cuanto a configuración de parámetros se refiere:

- **Antenas**, entre 2x2 y 4x4.
- **Esquema de modulación**, entre QPSK y 256QAM.
- **Balance prestaciones BER/*Throughput***, a través de la configuración del número de iteraciones. Entre 1 y 4 para detectores SESPA y entre 1 y 8 para los TESPA.
- **Salida**, *hard output/soft out.*

Los detectores han sido implementados, evaluados y comparados con los mejores publicados en la literatura especializada, ofreciendo como valor añadido, además de la alta configurabilidad, la posibilidad de decodificar 256QAM sin incrementar el área de ocupación. Esta característica es altamente competitiva con los detectores no lineales basados en KBest, que son muy sensibles, en cuanto a tasa de decodificación y área se refiere, con el esquema de modulación seleccionado.

Capítulo 5

Conclusiones y líneas futuras de trabajo

5.1. Conclusiones

Es esta tesis se ha desarrollado la primera arquitectura VLSI de baja complejidad computacional para el algoritmo SPA, cuya implementación es viable y competitiva frente a otras arquitecturas tanto lineales como no lineales. Para ello, en primer lugar, se ha realizado un estudio exahustivo de diferentes detectores, algoritmos y arquitecturas publicados en la literatura especializada para obtener mediante simulaciones, las prestaciones BER y la carga computacional alcanzadas por los mismos. Posteriormente se ha abordado, en profundidad, el estudio del algoritmo SPA y se han propuesto los mecanismos que hacen viable su implementación *hardware* para la obtención de las salidas *hard output* y *soft output*. Estos algoritmos han sido evaluados y comparados mediante simulaciones con otros publicados en la literatura. Finalmente se ha diseñado una arquitectura VLSI y, con ella, cuatro detectores con diferentes prestaciones y diferentes salidas (*hard output* y *soft output*).

Del estudio realizado sobre las prestaciones de los diferentes detectores MIMO existentes se desprende que sólo dos familias son realmente competitivas en lo que a implementación hardware se refiere, los detectores basados en cancelación sucesiva de interferencias (SIC) y los detectores basados en búsqueda en árbol. De entre los primeros, el detector VBLAST ofrece altas tasas de detección pero, sin embargo, sus prestaciones BER son limitadas

en comparación con los detectores básados en búsqueda en árbol. De entre los segundos, KBest, con tasa de detección fija y prestaciones BER cercanas a la ML se ha consolidado, en la literatura especializada, como la alternativa más viable para la implementación *hardware*, en detrimento de los detectores SIC.

La aparición del algoritmo de cancelaciones sucesivas (SPA) y la posibilidad de desarrollar un detector basado en el mismo se presenta como una alternativa razonable y competitiva, dentro de la familia de detectores SIC, frente a los detectores KBest. Sin embargo, la tarea de desarrollar un detector, cuya implementación *hardware* sea viable y competitiva, tomando como punto de partida el algoritmo original publicado, no es trivial. Los detectores basados en el algoritmo SPA ofrecen como valor añadido, frente a otros detectores SIC, la posibilidad de generar de forma iterativa distintas soluciones. Sin embargo, sus autores no establecen ningún método para controlar las repeticiones en las iteraciones, ni se estudia el número de iteraciones requeridas para aproximar la solución ML con un coste computacional razonable.

En este trabajo se proponen dos mecanismos de control de repeticiones, Simplied-ESPA y Table-ESPA, que hacen posible la implementación *hardware* del algoritmo SPA para la obtención de las salidas *hard output* y *soft output*. De la evaluación de los mismos se desprende que el primero, Simplied-ESPA, es útil durante las 4 primeras iteraciones. Es decir, sus prestaciones BER mejoran iteración a iteración hasta la cuarta, mientras que con el segundo mecanismo, Table-ESPA, esta situación se prolonga hasta la octava iteración, alcanzándose, en este caso, unas prestaciones BER cercanas a la ML. Para la obtención de estas dos salidas, *hard output* y *soft output*, de diferente naturaleza, se ha realizado la adaptación de los algoritmos existentes a los desarrollados en este trabajo.

Se ha desarrollado una arquitectura VLSI flexible, que se adapta a diferentes condiciones de transmisión y cumple con las últimas especificaciones publicadas en los estándares WiMAX y LTE. La flexibilidad de la arquitectura se resume en los siguientes aspectos:

1. Permite seleccionar diferentes configuraciones de antenas en transmisión y recepción, desde 2x2 hasta 4x4.

2. Permite seleccionar diferentes esquemas de modulación desde QPSK hasta 256QAM.

3. Permite seleccionar el balance entre entre tasa de transmisión y prestaciones BER/FER.

4. Ofrece las decisiones *soft output* y *hard output*.

Para el desarrollo de esta arquitectura se han diseñado los bloques de detección de símbolos y de procesado de salida utilizando la técnica de *pipeline-interleaving*. Mediante esta técnica se ha incrementado el *throughput* de los detectores en un factor 8. Se han ajustado las precisiones de los datos, consiguiendo unas pérdidas de implementación mínimas de 0,2 dB en cuanto a prestaciones BER se refiere, y se ha realizado un estudio detallado de las posibilidades de paralelización que ofrece la arquitectura. De esta manera, los detectores desarrollados pueden alcanzar tasas superiores a Gbps a costa, eso sí, de un incremento de área.

Se ha realizado la implementación de los detectores SESPA y TESPA, con salidas *soft output* y *hard output*, en los dispositivos FPGA y ASIC. Estos detectores han sido evaluados y comparados con los mejores publicados en la literatura especializada, consiguiendo la tasa de pico máxima de 465 Mbps para el detector SESPA 4x4 256-QAM, en un área de $3.83\,mm^2$ con una tecnología de $90\,nm$. Los detectores implementados ofrecen como valor añadido, además de la alta configurabilidad, la posibilidad de decodificar 256QAM sin incrementar el área. Esta característica es altamente competitiva con los detectores no lineales basados en KBest, que son muy sensibles, en cuanto a tasa de decodificación y área se refiere, con el esquema de modulación seleccionado. Además, los detectores basados en ESPA alcanzan unas prestaciones FER (*soft output*) claramente competitivas con los detectores KBest, debido a la mayor calidad del LLR generado por el ESPA. La comparación con otras arquitecturas flexibles seleccionadas demuestra que los detectores SESPA y TESPA ofrecen la mayor configurabilidad de parámetros de transmisión y el mejor equilibrio entre área, prestaciones BER y tasa detección.

5.2. Líneas futuras de trabajo

Se plantean tres grandes retos como continuación del trabajo presentado en esta tesis: mejora del *soft output*, mejora de la eficiencia del *hardware diseñado* y el diseño de una nueva arquitectura *hardware* para el algoritmo SPA.

La mejora del *soft output* se enmarca dentro de la parte algoritmica y surge del análisis que se ha desarrollado en esta tesis del algoritmo SPA, que se muestra altamente eficaz para la obtención de la salida *soft output*. De hecho, con pocas iteraciones las prestaciones FER son comparables a las obtenidas con KBest con elevadas K's. El cálculo del LLR según se explica en la sección 3.4 se realiza a partir de la métrica de los vectores candidatos, utilizándola como medida de la probabilidad de los bits correspondientes a los símbolos del vector. Forma parte de la naturaleza del algoritmo SPA el cálculo de un parámetro

de calidad asociado a los símbolos, el *base weight* (δ). Por lo tanto, una buena línea de investigación puede ser el desarrollo de una arquitectura especializada para la obtención del *soft output*, extrayendo información de la probabilidad de los bits a partir del cálculo de los δ.

La mejora de la eficiencia del *hardware* se enmarca dentro de la parte de implementación y surge a la vista de las ineficiencias que provoca el diseño actual. En la arquitectura presentada en esta tesis el bloque de detección de símbolos opera mediante 8 ramas en paralelo (Fig.A.1). En cada una de estas ramas se computan las unidades TBU y PSU. Cuando se obtiene el primer símbolo de un vector operan las 8 ramas, sin embargo, cuando se obtiene el segundo símbolo operan sólo 7 y así sucesivamente hasta la obtención del último símbolo que sólo opera una rama. Esto es una clara ineficiencia del *hardware* ocasionada por el funcionamiento en bucle de este bloque. El reto que se plantea en este punto es el aprovechamiento de las unidades TBU y PSU una vez detectado un símbolo en una rama determinada. Una posibilidad, podría ser la utilización de las unidades PSU para el cálculo de los base weight de segundo nivel que podrían ser útiles para el diseño de un control de repeticiones más preciso o para la mejora del *soft output*.

Del estudio topológico desarrollado en la sección 4.5 surge, también, una línea de trabajo para explorar las posibilidades de paralelización del *hardware* y realizar la implementación y evaluación de las mismas. En la misma sección se propone la realización de una estructura formada por dos bloques de detección de símbolos y uno de procesado de salida, que requiere ser modificado para la optimización del binomio área/*throughput*.

El diseño de una nueva arquitectura *hardware* es, sin duda, la propuesta más ambiciosa. Caben, en este punto, dos lineas de actuación. En primer lugar, el rediseño de la arquitectura actual teniendo en cuenta los avances de la tecnología en cuanto a dispositivos *hardware* se refiere. En la actualidad los dispositivos FPGA, por ejemplo, integran más multiplicadores y memorias que cuando se realizó el trabajo, lo que abre nuevas posibilidades de cara a una futura implementación.

En segundo lugar, el diseño de una nueva arquitectura tomando como punto de partida la serialización del bloque de detección de símbolos, que surge a la raíz de la ineficiencia del *hardware*, referente a la inactividad de las ramas TBU-PSU, ya comentada en el punto anterior. Una solución a este problema es el diseño de una arquitectura serie en árbol de este bloque, donde el número de ramas TBU-PSU va disminuyendo en cada etapa. De esta manera se provocaría un crecimiento eficiente del área sin penalizar el *throughput*, ya que la profundidad del *pipeline* quedaría multiplicada por 8 y no así el área.

Finalmente, cabe decir que ésta es la primera arquitectura desarrollada para el algoritmo SPA y por lo tanto, el margen de mejora es amplio como en cualquier otra primera propuesta en el ámbito de la ciencia y la tecnología.

Anexo A

Esquemas y señales de control

En esta sección se muestran los diagramas de trabajo utilizados durante la implementación de los detectores H-SESPA descritos en el capítulo 4, así como la tabla final de precisión de los datos ajustados para minimizar las pérdidas de implementación en cuanto a prestaciones BER se refiere.

Si bien la visualización de las mismas no es óptima en formato impreso en papel, se ha decidido incorporarlos para su visualización en formato digital (PDF), que permite su ampliación.

Cabe hacer la siguiente aclaración sobre la notación utilizada en estos diagramas: para la obtención de la matriz $\mathbf{G}$, inversa de la matriz de canal $\mathbf{H}$, en las implementaciones *hardware* se utiliza la descomposición $\mathbf{QR}$ para obtener $\mathbf{L} = \mathbf{R^{-1}}$. Si bien en esta tesis no se ha implementado esta descomposición ya que la fase de preprocesado no era objeto de trabajo, por aproximación a la realidad, sí se han utilizado las matrices $\mathbf{R}$ y su inversa $\mathbf{L}$, que son las que aparecen en la notación de los diagramas.

Los contenidos del anexo son los siguientes:

- Tabla de precisiones de los datos ajustadas para minimizar las perdidas de implementación de prestaciones BER de los detectores objeto de esta tesis. Las precisiones de los datos ($\mathbf{Qx}$) se han etiquetado en correspondencia con los esquemas A.1 y A.2 y son comunes a todas las implementaciones realizadas.

- Los esquemas del bloque de detección de símbolos y el bloque de pro-

Tabla A.1: Precisión de los datos establecida en la implementación de los detectores ESPA

		Señal	Total Bits	Bits Fraccionales
PSU	Q1	zf	18	7
	Q2	a	18	10
	Q3	div	18	5
	Q4	ROM in	16	5
	Q5	ROM out (dis)	16	11
	Q6	div out	18	10
	Q7	—	—	—
	Q8	δ	18	10
TBU	Q9	γR	18	10
	Q10	a	18	10
	Q11	div	18	5
	Q12	div out	18	10
	Q13	L	18	10
	Q14	y	18	10
MCU	Q15	γR	18	10
	Q16	dist	18	10
	Q17	R	13	10

cesado de salida del detector H-SESPA (detector SPA-Extendido con control de repeticiones simplificado y salida *hard output*). Estos esquemas contienen etiquetas con el nombre de las señales, referencia de cuantificación **Qx** y referencia de señal de control **CTx**. Estas referencias encuentran su correspondencia en la tabla de precisiones A.1 (**Qx**) y en el diagrama de señales de control A.2 (**CTx**).

- Cronograma de señales de control de detectores ESPA. Las señales de control (**CTx**) se han etiquetado en correspondencia con los esquemas A.1 y A.2 y son comunes a todas las implementaciones realizadas.

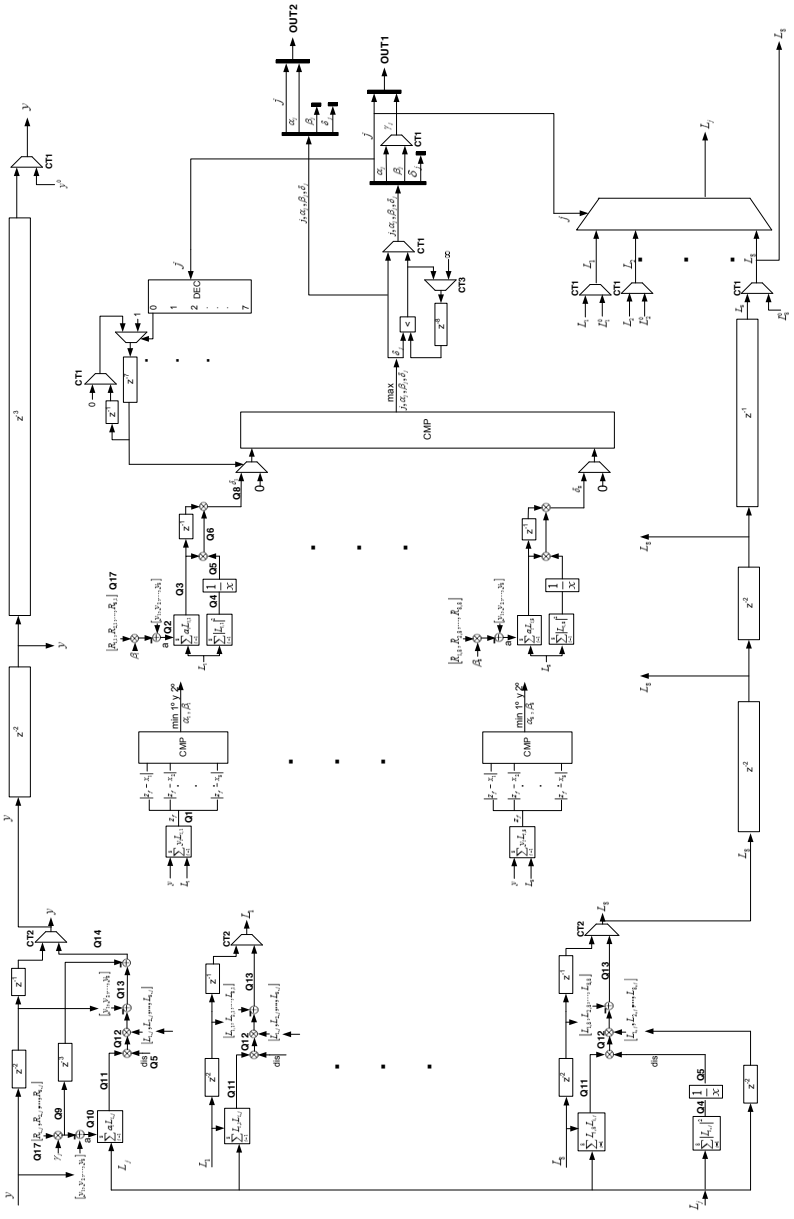


Figura A.1: Esquema con señales de control y cuantificación del bloque de detección de símbolos del detector H-SESPA

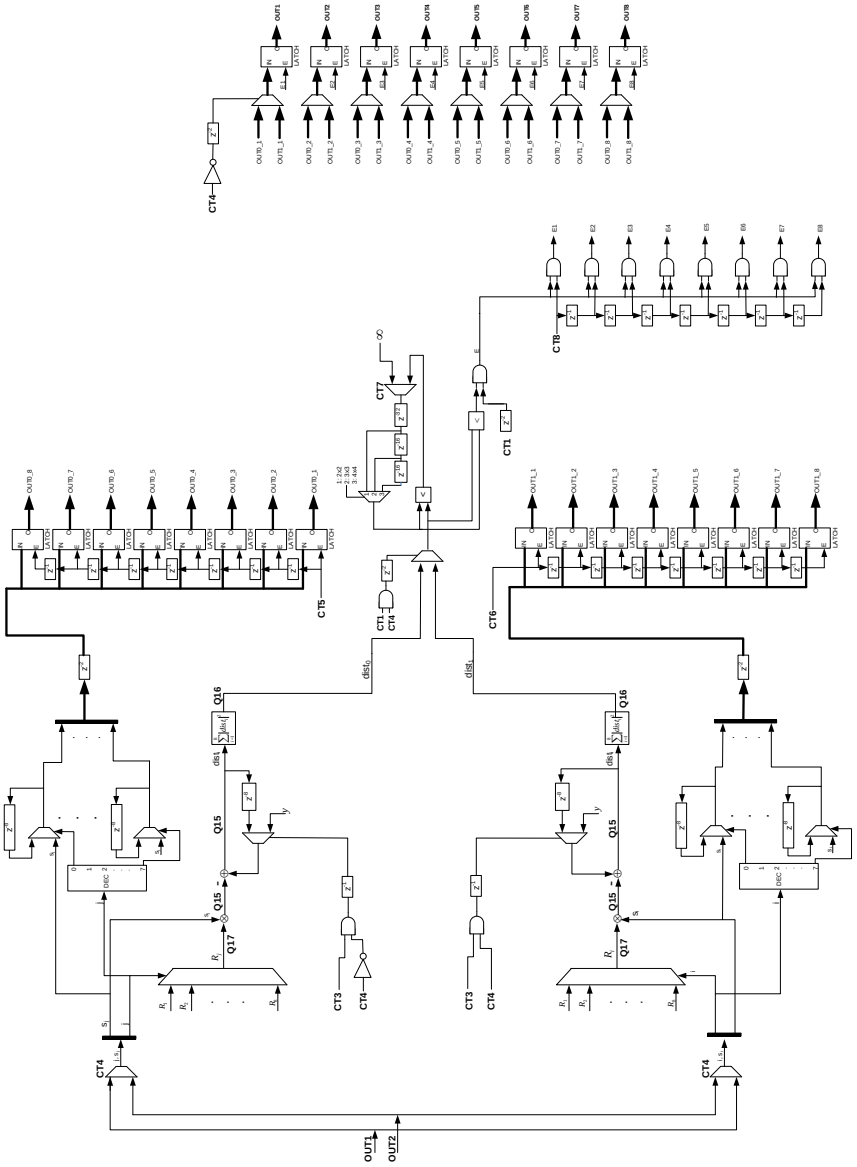


Figura A.2: Esquema con señales de control y cuantificación del bloque de procesado de salida del detector H-SESPA

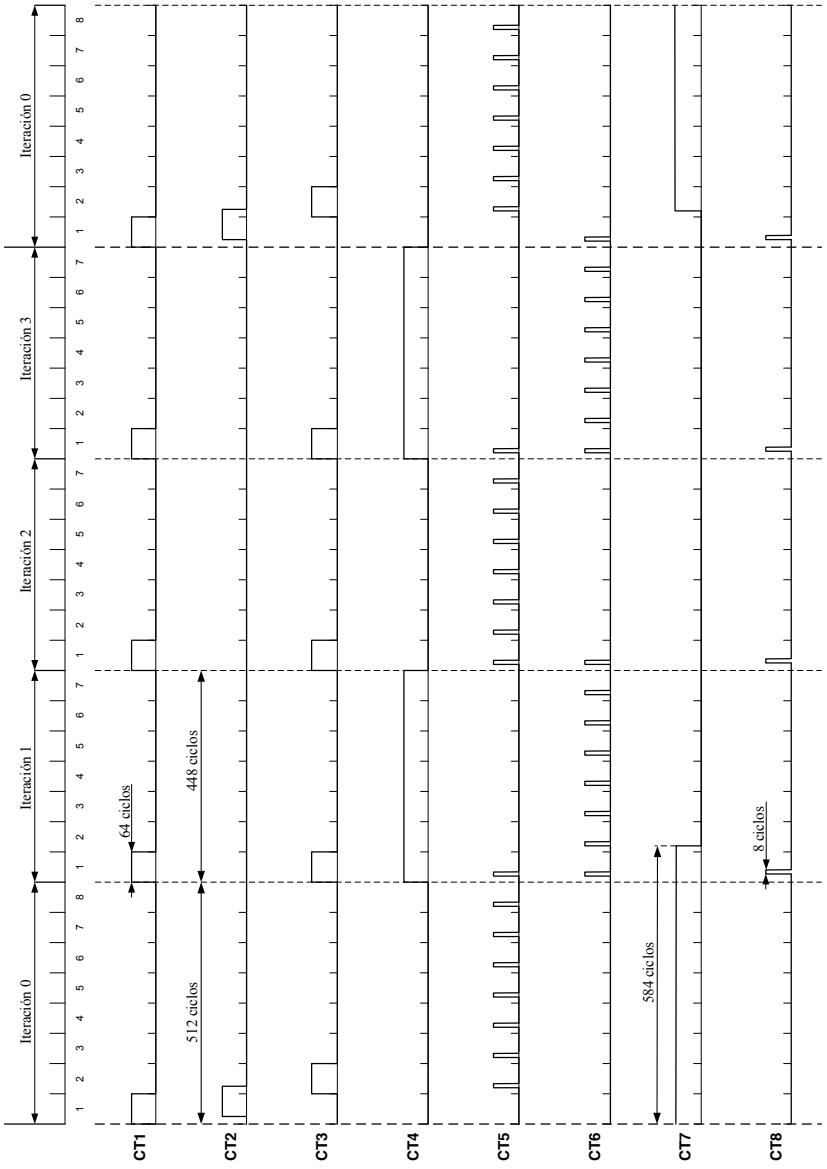


Figura A.3: Cronograma de señales de control de detectores ESPA

Bibliografía

[1] A. Paulraj, R. Nabar, and D. Gore, "Introduction to space-time wireless communications," Cambridge Univ. Press, Tech. Rep., 2003.

[2] P. Wolniansky, G. Foschini, G. Golden, and R. Valenzuela, "V-BLAST: an architecture for realizing very high data rates over the rich-scattering wireless channel," *Signals, Systems, and Electronics, 1998. ISSSE 98. 1998 URSI International Symposium on*, pp. 295–300, Sep-2 Oct 1998.

[3] E. Viterbo and J. Boutros, "A universal lattice decoder for fading channels," *Information Theory, IEEE Transactions on*, vol. 45, no. 5, pp. 1639–1642, July 1999.

[4] M. Pohst, "On the computation of lattice vectors of minimal length, successive minima and reduced basis with applications," *ACM SIGSAM Bull*, vol. 15, pp. 37–44, 1981.

[5] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Mathematics of Computation, Journal on*, vol. 44, pp. 463–471, Apr 1985.

[6] K. wai Wong, C. ying Tsui, R.-K. Cheng, and W. ho Mow, "A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels channels," *Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on*, vol. 3, pp. III–273–III–276 vol.3, 2002.

[7] J. Anderson and S. Mohan, "Sequential coding algorithms: A survey and cost analysis," *Communications, IEEE Transactions on*, vol. 32, no. 2, pp. 169 – 176, feb 1984.

[8] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *Information Theory, IEEE Transactions on*, vol. 48, no. 8, pp. 2201–2214, Aug 2002.

[9] C. Schnorr and M. Euchner, "Lattice basis reduction: Improving practical lattice basis reduction and solving subset sum problems," *Math. Programming*, vol. 66, pp. 181–191, 1994.

[10] H. Yao and G. Wornell, "Lattice-reduction-aided detectors for MIMO communication systems," *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, vol. 1, pp. 424–428 vol.1, Nov. 2002.

[11] A. Burg, "Vlsi circuits for mimo communicatios systems," Ph.D. dissertation, Swiss Federal Institute Ot Technology Zurich, 2006.

[12] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bolcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *Solid-State Circuits, IEEE Journal of*, vol. 40, no. 7, pp. 1566–1577, July 2005.

[13] M. Wenk, M. Zellweger, A. Burg, N. Felber, and W. Fichtner, "K-best MIMO detection VLSI architectures achieving up to 424 Mbps," *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pp. 4 pp.–1154, 2006.

[14] Z. Guo and P. Nilsson, "A 53.3 Mbps 4x4 16-QAM MIMO decoder in 0.35 μm CMOS," *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, pp. 4947–4950 Vol. 5, May 2005.

[15] Z. Guo and P. Nilson, "VLSI architecture of the soft-output sphere decoder for MIMO systems," *Circuits and Systems, 2005. 48th Midwest Symposium on*, pp. 1195–1198 Vol. 2, Aug. 2005.

[16] S. Chen, T. Zhang, and Y. Xin, "Relaxed K -Best MIMO Signal Detector Design and VLSI Implementation," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 15, no. 3, pp. 328–337, March 2007.

[17] C.-F. Liao, J.-Y. Wang, and Y.-H. Huang, "A 3.1 Gbps 8x8 Sorting Reduced K-Best Detector With Lattice Reduction and QR Decomposition," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 22, no. 12, pp. 2675–2688, Dec 2014.

[18] K. Su and I. J. Wassell, "A new ordering for efficient sphere decoding," *Communications, 2005. ICC 2005. 2005 IEEE International Conference on*, vol. 3, pp. 1906–1910 Vol. 3, May 2005.

[19] K. Su and I. Wassell, "Efficient MIMO detection by successive projection," *Information Theory, 2005. ISIT 2005. Proceedings. International Symposium on*, pp. 1681–1685, Sept. 2005.

[20] M. Mahdavi and M. Shabany, "Novel MIMO Detection Algorithm for High-Order Constellations in the Complex Domain," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 21, no. 5, pp. 834–847, May 2013.

[21] S. Mondal, A. Eltawil, C.-A. Shen, and K. Salama, "Design and Implementation of a Sort-Free K-Best Sphere Decoder," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 18, no. 10, pp. 1497–1501, Oct 2010.

[22] F. T. Corporation, *UMC 90nm logic SP (LowK) process standard core cell library.*

[23] B. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *Communications, IEEE Transactions on*, vol. 51, no. 3, pp. 389–399, March 2003.

[24] B. Steingrimsson, Z.-Q. Luo, and K. M. Wong, "Soft quasi-maximum-likelihood detection for multiple-antenna channels," in *Communications, 2003. ICC '03. IEEE International Conference on*, vol. 4, May 2003, pp. 2330–2334 vol.4.

[25] A. Lenstra, H. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," *Math. Ann.*, vol. 261, no. 4, pp. 515–534, 1982.

[26] M. Seysen, "Simultaneous reduction of a lattice basis and its reciprocal basis," *Combinatorica*, vol. 13, no. 3, pp. 363–376, 1993.

[27] D. Seethaler, G. Matz, and F. Hlawatsch, "Low-Complexity MIMO Data Detection Using Seysen's Lattice Reduction Algorithm," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. 2007 IEEE International Conference on*, vol. 3, Apr 2007, pp. 53–56.

[28] D. Seethaler and G. Matz, "Efficient vector perturbation in multi-antenna multi-user systems based on approximate integer relations," in *Proc. Eur. Signal Process. Conf.*, 2006, pp. 673–676.

[29] M. Shabany, A. Youssef, and G. Gulak, "High-Throughput 0.13 μm CMOS Lattice Reduction Core Supporting 880 Mb/s Detection," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 21, no. 5, pp. 848–861, May 2013.

[30] T. Kailath, H. Vikalo, and B. Hassibi, *Space-Time Wireless Systems: From Array Processing to MIMO Communications.* Cambridge University Press, 2005, ch. MIMO Receive Algorithms.

[31] C.-J. Huang, C.-W. Yu, and H.-P. Ma, "A Power-Efficient Configurable Low-Complexity MIMO Detector," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 56, no. 2, pp. 485 –496, feb. 2009.

[32] C. Studer, S. Fateh, and D. Seethaler, "Asic implementation of soft-input soft-output mimo detection using mmse parallel interference cancellation," *Solid-State Circuits, IEEE Journal of*, vol. 46, no. 7, pp. 1754 –1765, july 2011.

[33] D. Auras, R. Leupers, and G. Ascheid, "A novel reduced-complexity soft-input soft-output mmse mimo detector: Algorithm and efficient vlsi architecture," in *Communications (ICC), 2014 IEEE International Conference on*, June 2014, pp. 4722–4728.

[34] M. Damen, H. El Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *Information Theory, IEEE Transactions on*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.

[35] H. Kawai, K. Higuchi, N. Maeda, M. Sawahashi, T. Itoh, Y. Kakura, A. Ushirokawa, and H. Seki, "Likelihood function for QRM-MLD suitable for soft-decision turbo decoding and its performance for ofcdm mimo multiplexing in multipath fading channel," *IEICE TRANSACTIONS on Communications*, vol. E88-B, no. 1, pp. 47 – 57, jan. 2005.

[36] B. Hassibi, "An efficient square-root algorithm for BLAST," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. 2000 IEEE International Conference on*, vol. 2, 2000, pp. II737 –II740 vol.2.

[37] W. Forum, *WiMAX forum mobile system profile release 1.0 approved specification (revision 1.4.0:2007-05-02)*, WiMax Forum, May 2005.

[38] M. Shabany and P. Gulak, "A 675 Mbps, 4x4 64-QAM K-Best MIMO Detector in 0.13 μm CMOS," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 20, no. 1, pp. 135 –147, jan. 2012.

[39] C.-H. Yang and D. Markovic, "A Flexible DSP Architecture for MIMO Sphere Decoding," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 56, no. 10, pp. 2301 –2314, oct. 2009.

[40] C.-H. Liao, T.-P. Wang, and T.-D. Chiueh, "A 74.8 mW Soft-Output Detector IC for 8x8 Spatial-Multiplexing MIMO Communications," *Solid-State Circuits, IEEE Journal of*, vol. 45, no. 2, pp. 411 –421, feb. 2010.