Universidad de Alcalá
Escuela Politécnica Superior

GRADO EN INGENIERÍA ELECTRÓNICA DE COMUNICACIONES

Trabajo Fin de Grado

Diseño de una arquitectura en un Dispositivo FPGA para el Receptor de un Sistema de Posicionamiento Óptico

Autor: Laura Reyes Valbuena

Tutor/es: Álvaro Hernández Alonso

Junio de 2021

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

Grado en Ingeniería Electrónica de Comunicaciones

Trabajo Fin de Grado

Diseño de una arquitectura en un Dispositivo FPGA para el Receptor de un Sistema de Posicionamiento Óptico

Autor: Laura Reyes Valbuena

Tutor: Álvaro Hernández Alonso

TRIBUNAL:

Presidente: José Manuel Villadangos Carrizo

Vocal 1º: Pedro Alfonso Revenga de Toro

Vocal 2º: Álvaro Hernández Alonso

FECHA: Junio de 2021

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

En este Trabajo de Fin de Grado se pretende realizar una ampliación al sistema de posicionamiento de luz visible desarrollado en el Departamento de Electrónica de la Universidad de Alcalá. Se parte, por tanto, de una estructura física definida, es decir, el tipo de transmisor y receptor utilizado está fijado. Además, se conocen las características de los códigos que los transmisores emiten, así como los parámetros que el receptor obtiene de las señales captadas. También se cuenta con el algoritmo de posicionamiento que, a partir de los datos aportados por el receptor, consigue obtener las coordenadas del objeto a ubicar. Así, el presente documento muestra los pasos seguidos para la realización del diseño de una arquitectura hardware que permita implementar dicho algoritmo de posicionamiento en un dispositivo FPGA.

ABSTRACT

This Final Degree Project is intended to make an extension to the visible light positioning system developed at the Electronics Department from the University of Alcala. Therefore, the starting point is a defined physical structure, that is, the type of transmitter and receiver used is fixed. In addition, the characteristics of the codes that transmitters emit are known, as well as the parameters that the receiver obtains from the signals captured. There is also a positioning algorithm that, from the data provided by the receiver, manages to obtain the coordinates of the object to be located. Thus, this document shows the steps followed to carry out the design of a hardware architecture that allows implementing the aforementioned positioning algorithm in an FPGA device.

Resumen extendido

Actualmente, existen numerosas aplicaciones que requieren conocer la posición de objetos o personas tanto en áreas exteriores como interiores. Los sistemas de posicionamiento global, concretamente el GPS, es una de las tecnologías más usadas por su alta precisión y cobertura, pero están limitados a ambientes exteriores. En áreas interiores, dichos sistemas pierden exactitud y debido a la necesidad actual de ubicar objetos y personas en estas áreas, se han desarrollado los sistemas de posicionamiento local o LPS. Éstos ubican los móviles haciendo uso de diferentes tecnologías, como radiofrecuencia, infrarroja, óptica, etc.

Los sistemas de posicionamiento de luz visible (VLPS) constituyen una buena opción de LPS pues presentan una gran precisión, pese a que cuentan con algunos inconvenientes como la alta carga computacional o la reducida área de cobertura. Por ello, en [1] se propone el diseño de un VLPS basado en cuatro lámparas LEDs como transmisores y un Quadrant Photodiode Angular Diversity Aperture (QADA) como receptor. Utiliza una técnica de triangulación para estimar la posición del receptor por medio de un estimador de mínimos cuadrados (LSE). En este sistema propuesto, el algoritmo de posicionamiento desarrollado se ejecutaba en Matlab por lo que no se trataba de un sistema en tiempo real.

En el presente Trabajo de Fin de Grado se pretende dotar al sistema de posicionamiento mencionado de la capacidad de procesamiento en tiempo real, diseñando para ello una arquitectura hardware a implementar en una FPGA presente en la plataforma ZedBoard de la familia Zynq.

En primer lugar, se realizará un estudio del algoritmo de posicionamiento desarrollado en Matlab y representado en coma flotante. Debido a que la FPGA hace uso de la representación en coma fija, se va a ir analizando y transformando línea por línea dicho algoritmo a coma fija. Se estudiará el error de cuantificación cometido en cada caso, para poder determinar el número de bits óptimos necesarios para representar cada una de las señales del sistema con la mayor precisión posible.

En segundo lugar, teniendo en cuenta los anchos de palabra fijados para cada señal, se procede al diseño de la arquitectura en VHDL con la herramienta Vivado de Xilinx. De entre todas las operaciones que el algoritmo realiza, el punto clave es el diseño de un módulo capaz de correlar dos señales. Una vez realizado, se utilizarán IPs que ofrece Vivado junto con el diseño de un módulo comparador y una máquina de estados que se encargue de gestionar el sistema completo, para completar el diseño. Durante el diseño, irán realizándose simulaciones funcionales que vayan verificando el funcionamiento de los diferentes bloques que compondrán el sistema final.

En tercer y último lugar, se realizará un análisis de los resultados obtenidos al simular funcional y temporalmente el sistema completo. Se realizará una simulación funcional con las mismas señales experimentales capturadas de entornos reales, adecuando los valores, y se compararán los resultados obtenidos por la simulación con los dados por Matlab.

Capítulo 1

1. Introducción

El creciente desarrollo de aplicaciones y servicios, que requieren información acerca de la ubicación de un objeto o persona, ha provocado la necesidad de conseguir sistemas de posicionamiento precisos y asequibles. Los sistemas Globales de Navegación por Satélite (GNSS) cubren la mayoría de las necesidades de dichas aplicaciones, como son la amplia cobertura y alta precisión. Sin embargo, solo satisfacen estas necesidades en ambientes exteriores, pues en áreas interiores, efectos como la atenuación de las señales recibidas y el multicamino limitan la precisión del sistema. Todo esto ha hecho que los sistemas de posicionamiento en interiores se conviertan en un importante objeto de estudio hoy en día.

Los sistemas de posicionamiento en interiores, preferentemente, han de contar con un bajo coste y tamaño, larga vida útil, alta precisión y una amplia área de cobertura. Existen diferentes tipos de sistemas en función de la tecnología utilizada, donde cada uno de ellos presentan mejores cualidades en algunos aspectos y peores en otros. Así, los sistemas de posicionamiento acústicos pueden cubrir grandes áreas con bajos errores, pero les perjudica el efecto del multicamino y, además, la tasa de actualización suele ser baja. Por otro lado, los sistemas de posicionamiento por radiofrecuencia (Wi-Fi, RFID, BLE, Wi-Max…) cuentan con un elevado error debido a la dependencia que presentan a las fluctuaciones de las señales. Los sistemas mecánicos son pequeños y relativamente baratos, pero cuentan con una fuerte dependencia con la orientación del objeto y la aceleración, con importantes derivas y errores acumulativos a lo largo del tiempo. Por último, los sistemas ópticos basados en luz visible presentan precisiones en el rango de centímetros, tienen un bajo coste y son fáciles de integrar. En contraposición, cuentan con la desventaja del bajo volumen de cobertura que alcanzan.

Todo lo mencionado anteriormente, motivó el desarrollo del Trabajo de Fin de Máster descrito en [1], con el fin de poder ofrecer una alternativa realizable de un sistema de posicionamiento local. Éste se centra en la investigación de los sistemas de posicionamiento en interiores, profundizando en los sistemas ópticos basados en luz visible, con la intención de cubrir las limitaciones de los GNSS en ambientes interiores. Se implementa además un sistema de posicionamiento de luz visible (VLPS) simple y robusto, tridimensional, basado en cuatro lámparas LED como transmisores y un *Quadrant Photodiode Angular Diversity Aperture* (QADA) con una apertura cuadrada como receptor. Se propone un esquema de codificación para cada transmisión de los LEDs, una técnica de triangulación para estimar la posición del punto de incidencia en el receptor, un algoritmo que estima la posición final y una maximización del

volumen de cobertura de la celda, asegurando el cálculo de la posición con un número de balizas reducido.

El sistema propuesto en [1] registra las señales captadas por un osciloscopio para posteriormente tratarlas en un ordenador con los algoritmos desarrollados. Este procesamiento requiere tiempo y priva al sistema de la capacidad de trabajar en tiempo real. En el presente trabajo de fin de grado se pretende implementar en una FPGA el algoritmo desarrollado en [1] que permite estimar la posición del punto de incidencia en el receptor. Esto dota al sistema de la ausente capacidad de procesamiento en tiempo real mencionada y, además, permite una mayor libertad de movimiento al receptor, lo que podría aumentar a su vez, las posibles aplicaciones del sistema. Se propone una arquitectura especificada en VHDL, que reproduce la funcionalidad del algoritmo, siendo los objetivos parciales que permiten el correcto diseño de dicha arquitectura:

- Estudio y adecuación del algoritmo a implementar. Para dicha adecuación, ha sido necesario estudiar los conceptos de representación en coma flotante y en coma fija y el porqué de la necesidad de cuantificar las señales presentes en el algoritmo. También se ha hecho un estudio de las funciones presentes en la herramienta Matlab que han facilitado la tarea.
- Estudio del funcionamiento de los cores ofrecidos por Vivado, necesarios para el desarrollo de los módulos VHDL que constituirán la arquitectura.
- Diseño de un módulo capaz de implementar la correlación de dos señales.
- Integración con el resto de la algoritmia, definiendo arquitecturas eficientes que permitan maximizar el uso de los recursos a lo largo del tiempo. Análisis de soluciones paralelas y mixtas.
- Diseño de un bloque general que interconecte adecuadamente los módulos diseñados y que añada los cores necesarios para conseguir el cálculo de las coordenadas de los puntos de impacto en el receptor.
- Realización de simulaciones funcionales y temporales del diseño completo, así como de los módulos previamente diseñados y necesarios para el correcto funcionamiento de la arquitectura completa.

1.1. Contexto

El presente Trabajo de Fin de Grado está enmarcado dentro del Departamento de Electrónica de la Universidad de Alcalá, guardando una gran relación con la asignatura de Diseño Electrónico recibida durante el tercer año del Grado en Ingeniería Electrónica de Comunicaciones. Se ha hecho uso de los conocimientos básicos sobre VHDL y la herramienta de diseño Vivado de Xilinx aprendidos en ella, para el estudio y comprensión de los diferentes cores ofrecidos por dicha herramienta, así como para el desarrollo general de la arquitectura. A su vez, la línea de investigación sobre sistemas locales de posicionamiento, así como el diseño de arquitecturas eficientes, forman parte primordial del grupo de investigación GEINTRA, dentro del cual se ha integrado el desarrollo de este trabajo.

1.2. Estructura de la memoria

Los objetivos anteriormente mencionados, se han ido completando a lo largo del desarrollo del trabajo. Se ha plasmado el proceso en esta memoria, desde el estudio del estado del arte de los

sistemas de posicionamiento hasta las conclusiones finales obtenidas de la arquitectura VHDL propuesta. A continuación, se describe la estructura del presente documento:

1. **Introducción al trabajo de fin de grado**. En este apartado se realiza una breve contextualización del proyecto, explicando cuál es la motivación de su realización, es decir, cuál es la problemática que pretende resolverse y el porqué de esa necesidad.

2. **Antecedentes**. En esta sección se expone la situación actual de los sistemas de posicionamiento y los tipos de sistemas que hay en función de la tecnología utilizada. Se hace especial hincapié en los sistemas de posicionamiento ópticos basados en luz visible, por ser el sistema diseñado en [1] y objeto de estudio principal de este trabajo un sistema de este tipo. Además, se mencionan diferentes técnicas de posicionamiento. Por otro lado, se realiza una breve explicación acerca de las FPGAs, centrándose especialmente en la ZedBoard, que es el dispositivo utilizado en este trabajo.

3. **Descripción del sistema global**. La primera parte de este punto presenta de forma general el sistema de posicionamiento desarrollado en [1], centrándose en las partes que han sido claves para el entendimiento de la problemática a resolver: el desarrollo del algoritmo de obtención de los puntos de impacto. En la segunda parte, se ha explicado el proceso seguido para la adecuación de dicho algoritmo, realizando la comparación de las señales en coma flotante y en coma fija con el fin de poder determinar el número de bits necesarios para la representación de cada una de las señales del sistema en la FPGA.

4. **Descripción de la arquitectura propuesta**. En este apartado se explican paso a paso los razonamientos seguidos para la elección de la estructura de la arquitectura propuesta, así como para el diseño de los diferentes módulos VHDL. Se muestra también, el funcionamiento de dichos módulos por medio de diagramas de bloques, flujogramas y capturas de las simulaciones realizadas en el entorno de desarrollo Vivado.

5. **Resultados experimentales**. En este punto se muestra por un lado el comportamiento real de la arquitectura diseñada, a través de simulaciones temporales, ante señales de entrada diferentes a las experimentales, definidas y utilizadas en [1]. Por otro lado, se muestran los resultados obtenidos mediante la realización de simulaciones funcionales, inyectando esta vez a la entrada las señales experimentales utilizadas en [1].

6. **Conclusiones y trabajos futuros**. Se exponen las conclusiones obtenidas del trabajo realizado tras el análisis de los resultados parciales y generales del diseño elaborado. Además, se hace una breve reflexión acerca de las posibles mejoras del diseño y de posibles trabajos futuros.

Capítulo 2

2. Antecedentes

En este punto, se van a introducir una serie de conceptos básicos que ayudarán a poner en contexto este trabajo y a su mejor comprensión. Se comentarán, por un lado, diferentes tipos de sistemas de posicionamiento existentes, junto con las técnicas de posicionamiento más utilizadas. Por otro lado, se describirán algunos de los dispositivos que, actualmente, permiten el procesamiento digital de señales y se indicará cuál de ellos se va a utilizar en este trabajo y porqué.

2.1. Sistemas de posicionamiento

El propósito de los sistemas de posicionamiento es localizar un objeto o persona en un espacio, ubicándolo después respecto a unas referencias. Existen dos grandes tipos de sistemas de posicionamiento en función del espacio donde se encuentre el móvil a ubicar:

- Los **sistemas de posicionamiento globales** obtienen las coordenadas del objeto respecto a marcos de referencias globales. Por lo tanto, se utilizan cuando se desea conseguir la ubicación de un objeto situado en un ambiente exterior, como puede ser el parque de una ciudad o una calle.
- Los **sistemas de posicionamiento locales** adquieren la ubicación del objeto respecto a marcos de referencia local, como puede ser una habitación o un área de metros cuadrados. Puede intuirse entonces que estos sistemas se utilizan cuando se quiere posicionar con precisión un objeto presente en un área reducida interior, como puede ser ubicar a una persona en el interior de una vivienda.

2.1.1. Sistemas de posicionamiento en exteriores

Los sistemas de posicionamiento en exteriores, o sistemas globales de navegación por satélite (GNSS), brindan posicionamiento geoespacial con una precisión de unos pocos metros, aportando valores de latitud, altitud y longitud, mediante una constelación de satélites y señales de radio transmitidas a lo largo de un LoS.

El GNSS más utilizado es el Sistema de Posicionamiento Global (GPS) [2], desarrollado e implementado por el Departamento de Defensa de Estados Unidos. Otros sistemas de navegación por satélite son el GLONASS [3], desarrollado por la Unión Soviética, administrado a

día de hoy por la Federación Rusa y utilizado por su Ministerio de Defensa. El GNSS BeiDou-3 [4], desarrollado en China, espera ser a partir de 2020 un sistema alternativo al GPS, el GLONASS y el GNSS de la Unión Europea denominado Galileo [5], ofreciendo una mayor precisión en el rango de los milímetros.

2.1.2. Sistemas de posicionamiento local (LPS)

Los sistemas globales de navegación por satélite no son una buena opción para el posicionamiento en ambientes interiores debido a que la precisión empeora drásticamente por el efecto del multicamino y la atenuación de la señal [6]. Principalmente ocurre en el interior de viviendas con varios pisos, paredes gruesas y en zonas alejadas de las ventanas. Es por esto que se necesitan otras técnicas que cubran estas limitaciones.

Son los sistemas de posicionamiento local los basados en tecnologías que permiten la localización de objetos o personas situadas en áreas interiores con una precisión suficientemente aceptable. A diferencia de los sistemas de posicionamiento global, los LPS no cuentan con una sola técnica que constituya todos los sistemas, sino que existen diferentes tecnologías que dan lugar a diferentes tipos de LPS con sus propias ventajas e inconvenientes. El patrón común que se cumple en todos los LPS es la existencia de transmisores, dentro del área interior a cubrir, que emiten balizas orientadas a un tipo de tecnología (radiofrecuencia, acústica…), las cuales serán utilizadas para obtener una información concreta en el receptor en función del tipo de técnica de posicionamiento utilizada. En la Figura 1 se muestran los principales sistemas de posicionamiento junto con la precisión y cobertura que brindan.

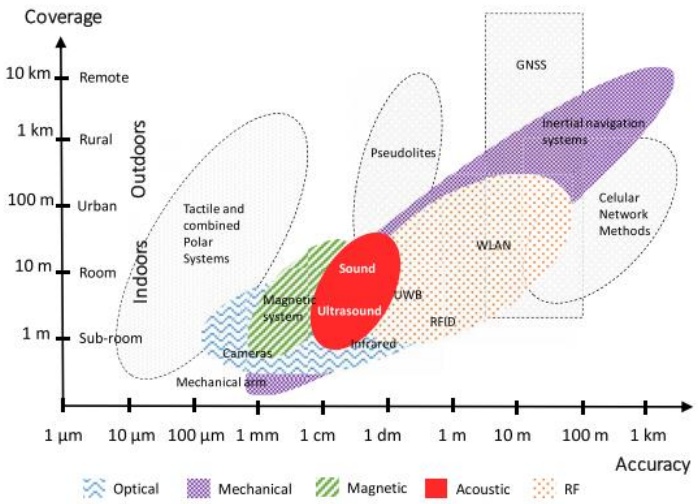

Figura 1. Cobertura y precisión de los sistemas de posicionamiento más relevantes [7].

A continuación, se describen los LPS más relevantes hoy en día junto con las técnicas de posicionamiento existentes.

2.1.2.1. Radio Frecuencia

Es una de las tecnologías más usadas en estos sistemas. Permiten adquirir la posición haciendo uso de cualquier señal de radio, por lo que de esta tecnología deriva la posibilidad de implementar un sistema de posicionamiento basado en Wi-Fi, Ultra Wide Band (UWB), Identificación por Radio Frecuencia (RFID), etc. Estos sistemas son una buena opción, pues

permiten cubrir grandes áreas por la capacidad de estas señales de atravesar paredes y por la posibilidad de evitar la instalación de nuevos sistemas, si se utiliza la tecnología Wi-Fi o Bluetooth.

2.1.2.2. Ultrasonidos

Por lo general, los sistemas de posicionamiento basados en ultrasonidos constituyen un recurso simple, barato y efectivo, ya que son capaces de posicionar un objeto con un error de unos pocos centímetros. Uno de sus inconvenientes es la pequeña área de cobertura que cubren y la insuficiente robustez frente a perturbaciones externas. Debido a la baja velocidad de propagación de la señal ultrasónica, estos sistemas pueden estimar con suficiente precisión el tiempo de vuelo de la señal o la diferencia de tiempo de llegada. Es decir, suelen utilizar medidas para el posicionamiento como TDoA (Time Difference of Arrival) o TDA (Time of Arrival).

2.1.2.3. Infrarrojos

La radiación infrarroja se localiza aproximadamente entre los 700nm y los 300µm, por lo que las señales IR se reflejan en paredes y demás obstáculos que pueda haber en una habitación, generando el efecto negativo de multicamino. Esto ha limitado la utilización de los sistemas de infrarrojo a aquellas áreas interiores en las que se pueda asegurar una línea de visión directa entre transmisor y receptor. Pero también ha impulsado la investigación de otras alternativas, como por ejemplo el análisis de este efecto en función de la utilización de diferentes técnicas de posicionamiento [9].

2.1.2.4. Sistemas de posicionamiento local visible (VLPS)

Estos sistemas de posicionamiento son utilizados tanto para ambientes exteriores como interiores, pero la gran mayoría de las veces se utilizan para áreas interiores debido a la gran influencia de luz ambiente que sufren en el exterior. Para el posicionamiento en interiores, comúnmente, se utiliza una matriz de diodos o un solo diodo LED como transmisor mientras que los receptores pueden estar basados en fotodetectores simples y sensores de imagen.

Los sistemas de posicionamiento con receptores basados en sensores de imagen a menudo utilizan cámaras CMOS [15][16] o la cámara de los teléfonos móviles inteligentes [14]. Sin embargo, precisan de algoritmos de procesamiento de imágenes complejos que dificultan el cálculo de la ubicación del objeto [17].

Por otro lado, los sistemas de posicionamiento basados en fotodetectores, ya sean fotodiodos o una matriz de ellos, producen una corriente proporcional a la intensidad de luz incidente en dicho receptor. A partir de la corriente y mediante un algoritmo de posicionamiento se puede estimar la posición del receptor. Existen dos sensores ampliamente utilizados en VLPS: el Detector Sensible a la Posición (PSD) y el *Quadrant Photodiode Angular Diversity Aperture (QADA)* [*18*], siendo este último el utilizado en el desarrollo del sistema en [1].

2.1.2.5. Técnicas de posicionamiento

Por último, las técnicas de posicionamiento más frecuentes son la triangulación y trilateración, desarrolladas a partir de medidas de:

- **Tiempos de Llegada** (ToA) o **Diferencias de Tiempo de Llegada** (TDoA).
    - ToA mide el tiempo absoluto que tarda la señal en incidir en el receptor.

  - TDoA mide la diferencia de tiempo entra las señales incidentes en el receptor.

Ambas técnicas cuentan con una serie de desventajas: por un lado, para ToA es necesaria una correcta sincronización entre transmisor y receptor; y por otro, el elevado valor de la velocidad de la luz provoca errores elevados. En ambos casos, la distancia se calcula a partir de la velocidad de la luz en el medio, la longitud de onda de la señal transmitida y el tiempo medido, entre transmisor y receptor, con ToA o TDoA. Al ser el valor de la velocidad de la luz tan elevado, un error de estimación del orden de los nanosegundos se convierte en un importante error de posicionamiento.

- **Intensidad de la señal recibida** (RSS): esta técnica mide la potencia de la señal recibida. Deben tenerse en cuenta por tanto los posibles reflejos y el efecto del multicamino, lo que incrementa la complejidad de los algoritmos. Es por ello que esta técnica suele utilizarse en VLPS que cubran distancias cortas.
- **Ángulos de llegada** (AoA), para utilizar esta técnica es necesario desarrollar un algoritmo que estime los ángulos de incidencia de las balizas en la superficie del receptor. En este caso, las fluctuaciones de las señales no afectan a la precisión del sistema y, además, no es necesario sincronizar las fuentes. Sin embargo, una pérdida del LoS entre transmisor y receptor degradará la precisión del sistema. Además, serán necesarias al menos tres balizas para obtener una localización en un espacio tridimensional.

2.2. Sistemas electrónicos digitales

Para el diseño e implementación de sistemas de control electrónico, procesamiento de señales digitales, implementación de un algoritmo de posicionamiento para un VLPS, que es el caso abordado en este trabajo de fin de grado, etc., existen varias opciones en el mercado. Éstas pueden dividirse en dos grandes grupos: los sistemas electrónicos digitales programables y los configurables.

2.2.1. Sistemas programables

Un sistema programable es un circuito integrados que contienen un microprocesador, el cual ha de ser programado por software para realizar una funcionalidad concreta según la necesidad del usuario. Se trata de sistemas baratos y flexibles, pues pueden utilizarse para multitud de tareas sin necesidad de contar con sistemas específicamente diseñados para una aplicación.

Existen dos sistemas destacables, que pueden ser utilizados para la implementación de algoritmos y tratamiento digital de señales:

- Los **microcontroladores** son dispositivos de bajo coste que contienen, además de una unidad de procesamiento o CPU, elementos de memoria, periféricos y puertos de E/S.
- Los **procesadores digitales de señales** o **DSP** (Digital Signal Processor) se implementan mediante microprocesadores expresamente diseñados para favorecer el cálculo rápido de las operaciones básicas asociadas al procesado digital de señales. Estas son la multiplicación y acumulación de dos secuencias o producto acumulativo, que son conjuntamente conocidas como operación MAC. Son capaces de ejecutar dichas operaciones a grandes velocidades gracias también a la arquitectura de memoria, que les permite obtener instrucciones y datos a procesar al ritmo que la CPU se los manda. Es decir, presentan un acceso múltiple que posibilita cargar de forma simultánea varios

operandos. Además, cuentan con un conjunto de periféricos que les permite comunicarse con el resto de los componentes del sistema.

Debido a la gran flexibilidad que presentan los DSPs y a que, justamente, están especializados en realizar la operación principal del algoritmo de posicionamiento que se pretende implementar, podría concluirse que éste es el dispositivo ideal donde desarrollar dicho algoritmo. Pero lo cierto es que las FPGAs actuales pueden contar con todos los bloques aritméticos, arquitecturas y memorias necesarias para imitar el funcionamiento de los DSPs. Además, cuando esto se combina con la paralelización permisible por las FPGAs, se tienen sistemas que aumentan su rapidez en un factor de 500 o más [19], respecto a los DSPs desarrollados con microprocesadores genéricos.

2.2.2. Sistemas lógicos configurables

Los sistemas lógicos configurables cuentan con una estructura interna dinámica que permite, a través de un software, modificar su configuración tantas veces como se desee, consiguiendo así crear arquitecturas hardware que se ajusten a las exigencias de una aplicación. Por tanto, la principal y crucial diferencia entre los sistemas programables y los configurables es que, los programables no modifican nunca su arquitectura hardware mientras que los configurables basan su funcionamiento en este hecho. Esto les permite paralelizar operaciones, pues pueden, por ejemplo, implementar las arquitecturas de dos microprocesadores y trabajar a la vez, simulando el funcionamiento que tendrían dos microprocesadores físicamente separados.

2.2.2.1. Evolución de la lógica configurable

El primer dispositivo configurable, **PROM** (Programmable logic array), se presenta en 1970 y está constituido por un plano fijo de puertas AND y otro configurable de puertas OR. Desempeñaban la funcionalidad de memorias para computadores y facilitaban la implementación de reducidos bloques combinacionales y LUT, mientras que los procesos síncronos no podían llevarse a cabo por la ausencia de Flip-Flops.

Después, los **PLAs** (Programmable logic array), implementados en Signetics por Ron Cline en 1975, estaban constituidos por dos planos configurables donde las puertas AND y OR se compartían. Eran un poco más lentos que los PROM y al igual que ellos, no permitían los procesos síncronos.

Los dispositivos **PAL** (Programmable array logic) se desarrollaron en MMI en 1978 y estaban compuestos por un plano configurable AND y un plano fijo OR. Era más rápida que la PLA, de fácil uso y solo permitía implementar lógica sencilla y no realizaba un uso eficiente de los recursos.

Posteriormente, en 1984 y de la mano de ALTERA saltó al mercado el primer **CPLD** (Complex PLD), los cuales se configuraban mediante la escritura de una memoria FLASH o EEPROM interna. En este caso, los CPLD sí hacían un uso eficiente de sus recursos, estaban orientados a diseños de dificultad media y necesitaban de un ordenador personal para su configuración.

Finalmente, en 1984 Xilinx desarrolla la primera **FPGA.** Mantenía el uso eficiente de los recursos, permitía desarrollar diseños con lógica muy compleja y con una alta frecuencia de trabajo. Además, necesitaban de una herramienta de software orientada a la creación de sistemas electrónicos llamada EDA (Electronic Design Automation).

A continuación, por ser la tecnología elegida para la implementación del algoritmo de posicionamiento, se va a describir con mayor profundidad la estructura de una FPGA.

2.2.2.2. Dispositivos FPGAs

Las FPGAs (Field Programmable Gate Arrays) son dispositivos electrónicos basados en semiconductores y son reconfigurables, constituidos principalmente por una matriz de bloques lógicos configurables o CLBs (*Configurable Logic Blocks*). Estos bloques pueden conectarse entre sí y con otros bloques de entrada/salida o IOBs (*Input/Output Block*) mediante la programación de un tercer bloque llamado *switch*. En la Figura 2 se muestra la distribución interna de una FPGA, donde pueden observarse los bloques mencionados anteriormente.

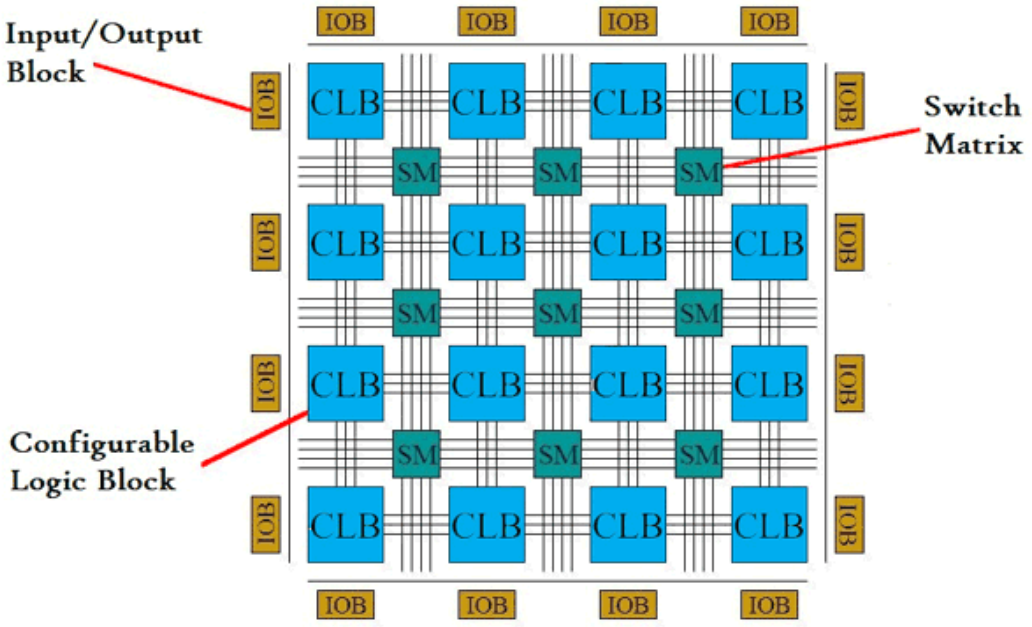

Figura 2. Estructura interna simplificada de una FPGA [20].

Los CLBs implementan las funciones lógicas necesarias mediante un registro, una LookUp Table (LUT) y un multiplexor, aunque en las nuevas familias se agrupan en un solo CLB varios de estos elementos. En la Figura 3 se muestra un bloque lógico configurable básico.

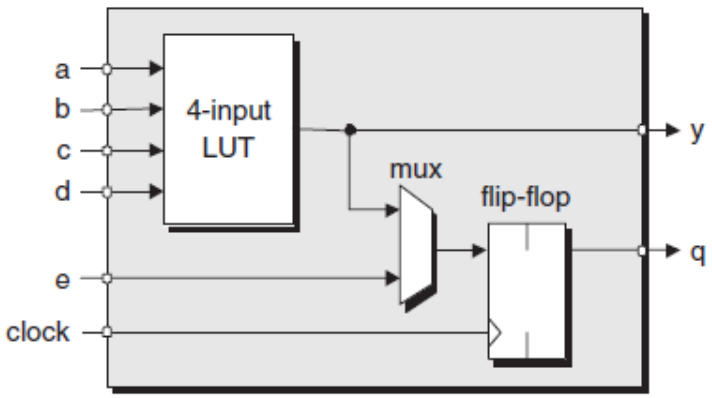

Figura 3. Estructura interna de un CLB básico[19].

Un registro o Flip-Flop es un biestable tipo D que logra estabilizar un nivel entre flancos de subida o bajada de la señal de reloj. Es por ello que estos bloques, y por tanto las FPGAs, pueden llevar a cabo procesos síncronos además de combinacionales. Por otro lado, las LUTs son memorias que contienen las tablas de verdad de cualquier función lógica, lo que permite, en función de su configuración, llevar a cabo cualquier diseño lógico. En la Figura 4 se muestra un ejemplo de cómo se configuraría una LUT para un circuito lógico dado. Puede observarse además cómo es un bloque SRAM el que se encarga de realizar dicha configuración.

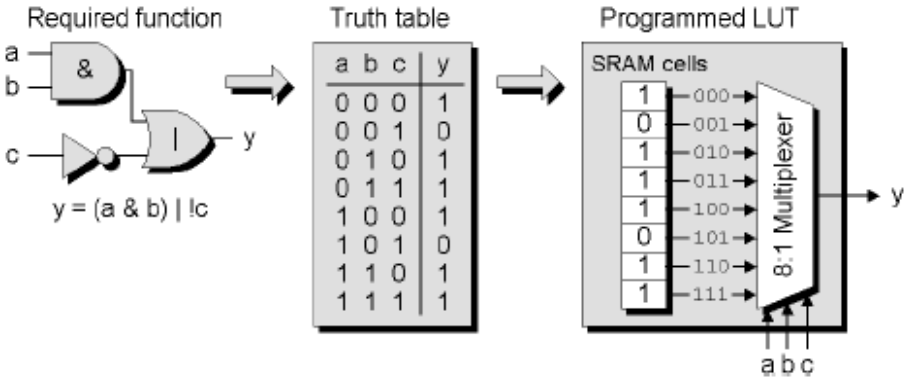

Figura 4. Ejemplo de configuración de una LUT a partir de una función lógica dada [19].

En la Figura 4 se ha mostrado la estructura de un CLB básico pero, dependiendo de la familia base a la que pertenezca la FPGA y su modelo, existen diferentes tipos de CLB. De hecho, actualmente, los CLBs están constituidos por *slices*, estando estos a su vez constituidos por celdas lógicas (LC). Cada fabricante, siendo los más relevantes Xilinx, Altera, fija sus propios nombres para cada bloque. Así, Xilinx entiende por LC lo que anteriormente se representó como CLB y estructura los CLBs en *slices*. Dependiendo de la familia a la que pertenezca una FPGA, los CLBs que la constituyen pueden estar compuestos por más o menos *slices*. En la Figura 5 se muestra un esquema que ejemplifica esta estructura.

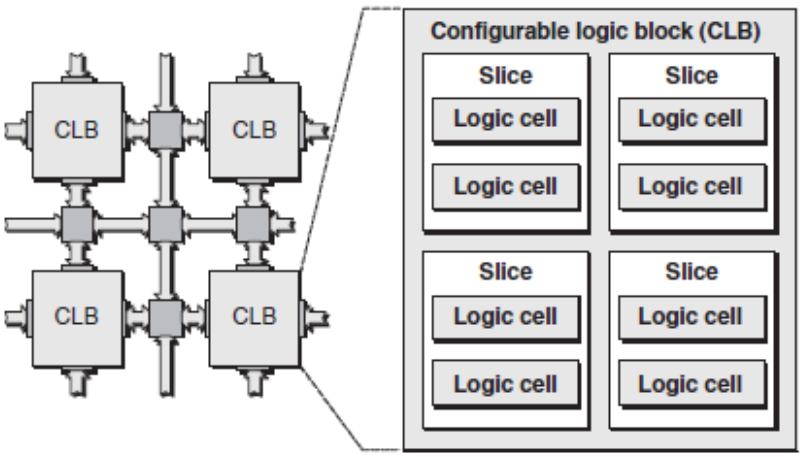

Figura 5. Estructura interna de un CLB con cuatro slices [19].

Además, en los últimos años las FPGAs han incorporado bloques para mejorar las prestaciones como por ejemplo DSPs, que se encargan de realizar las sumas y multiplicaciones, operaciones

MAC y demás operaciones aritméticas, bloques de memoria RAM, procesadores de propósito general, enrutados entre bloques, etc.

2.2.3. Systems-on-Chip (SoC)

Un System on Chip o SoC está compuesto por un solo chip de Silicio capaz de implementar la funcionalidad de un sistema completo, sin necesidad de contar con diferentes chips físicos. Estos sistemas combinan las características inherentes a un sistema digital programable y a un sistema con lógica configurable. Las prestaciones de estos sistemas podrían conseguirse acoplando dispositivos físicamente separados en una placa de circuito impreso (PCB), pero la opción de los SoC permite una transferencia de datos más rápida y fiable entre los diferentes elementos del sistema, menor coste, menor consumo de energía y menor tamaño físico.

Antiguamente, se entendía un SoC como un circuito integrado de aplicación específica o ASIC. Los ASICs son circuitos específicos para una aplicación, por lo que su lógica es desarrollada a medida para cubrir todas las necesidades de dicha aplicación y no sirven para otra. Por ello, su implementación conlleva un alto coste y son factibles para producciones de gran volumen.

Para el desarrollo de este trabajo de fin de grado se ha elegido la tarjeta de desarrollo ZedBoard de Xilinx, que cuenta con un SoC de la familia Zynq. Como puede observarse en la Figura 6, además del SoC la tarjeta cuenta con una gran cantidad de periféricos. Nótese en cualquier caso que no se hará uso del procesador, sino solamente de la lógica configurable, a lo largo de este trabajo.

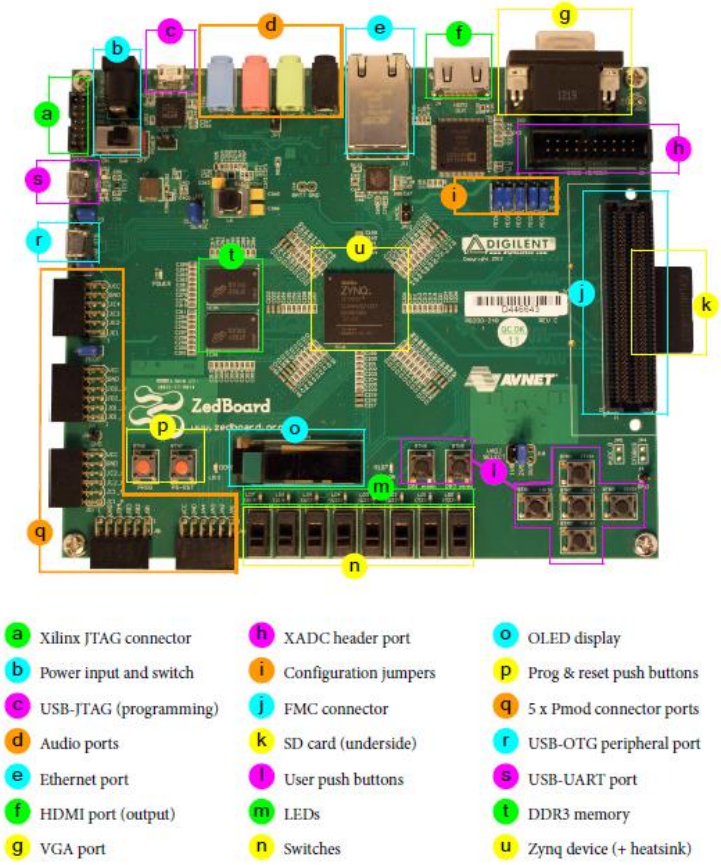

Figura 6. Tarjeta de desarrollo ZedBoard [22].

Concretamente, la ZedBoard contiene un SoC Zynq-ZC7Z020, uno de los dispositivos más pequeños de la gama Zynq-700. Combina un procesador ARM Cortex-A9 de doble núcleo con la estructura lógica de una FPGA. Concretamente se basa en la arquitectura de la FPGA Artix-7 de Xiliinx. Cuenta con una capacidad de 13300 slices lógicos, 220 DSP48E1s y 140 BlockRAMs [22]. Además, presenta interfaces AXI que proporcionan conexiones de gran ancho de banda y baja latencia entre las dos partes del dispositivo, conocidas ambas como PS (Processing System) y PL (Programmable Logic). En la Figura 7 se muestran ambas áreas junto con los periféricos asociados.

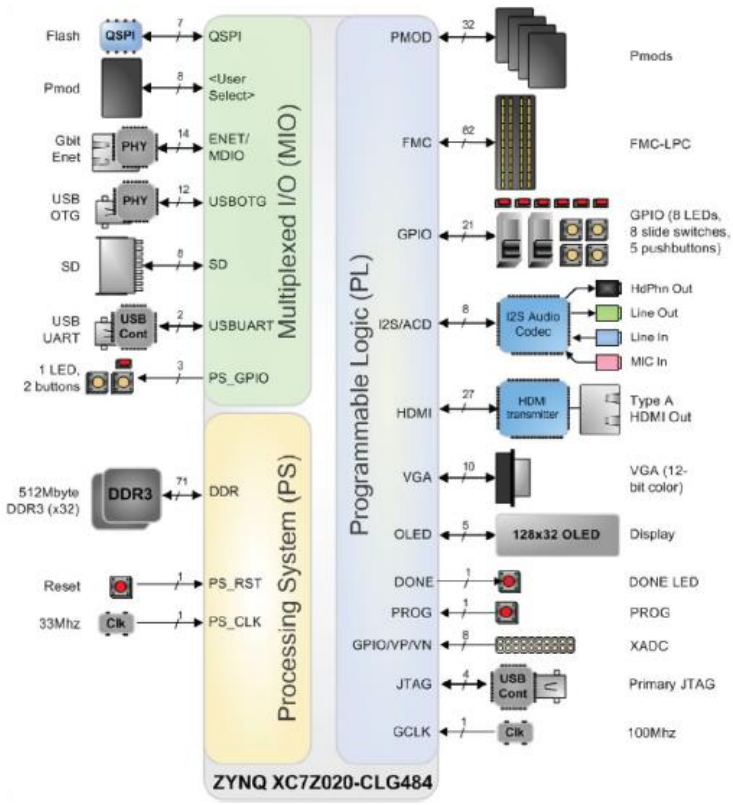

Figura 7. Diagrama de bloques de la tarjeta de desarrollo ZedBoard [21].

Este tipo de sistemas permite utilizar cada área de forma independiente, pudiendo implementar cada etapa del algoritmo en la parte que mejor resultado proporcione, en lo que a tiempo de procesamiento y eficiencia se refiere.

Capítulo 3

3. Descripción del sistema global

3.1. Sistema local de posicionamiento previo

El sistema de posicionamiento local basado en luz visible (VLPS), objeto de estudio de este TFG y para el cual se desarrolla la arquitectura a implementar en una FPGA, ha sido estudiado, analizado y desarrollado previamente en [1]. Está constituido por cuatro focos LEDs como transmisores y un *Quadrant Photodiode Angular Diversity Aperture* (QADA) como receptor. Además, el receptor QADA tiene una apertura que ensombrece partes del mismo, dependiendo de la posición del emisor. Cada uno de los cuatro transmisores emite una secuencia distinta con un esquema de codificación basado en secuencias LS de 1151 bits, con una modulación *Binary Phase Shift Keying* (BPSK). Esta técnica de codificación dota al sistema de capacidad de acceso múltiple y una mayor robustez frente a una SNR baja, entre otras.

Para estimar la posición del receptor, el sistema sigue los siguientes pasos. En primer lugar, con una técnica de triangulación basada en *Non-Linear Least Squares* (NLS), utilizando el método de Gauss-Newton, obtiene el punto de incidencia en el receptor. En segundo y último lugar, mediante un *Least Squares Estimator* (LSE) y algunas consideraciones geométricas se obtiene la posición local del receptor (en el volumen propuesto).

El sistema fue validado mediante simulación y pruebas experimentales para distintas posiciones en una celda de 2 x 2 x 2 m, obteniéndose errores de posicionamiento por debajo de los 6 cm. La estructura completa del sistema se muestra en la Figura 8.

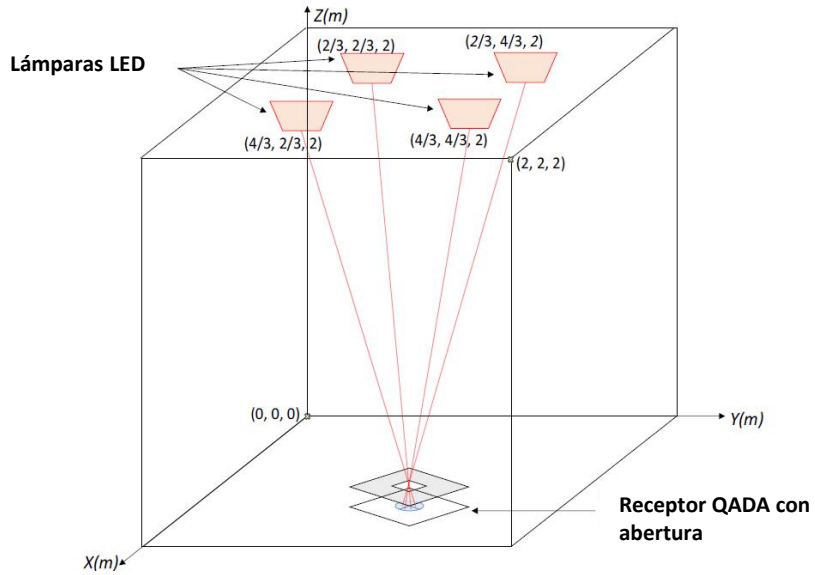

Figura 8. Sistema de posicionamiento con cuatro emisores LED y una matriz de fotodiodos con una apertura cuadrada en una celda unitaria. (Tener en cuenta que los elementos no están a escala) [1].

3.1.1. Transmisores

El transmisor propuesto para el sistema de posicionamiento consta de cuatro LED_i, compuestos cada uno de ellos por cinco módulos OVM12F3W7 conectados en paralelo, teniendo cada módulo 3 LEDs en serie. Estos transmisores son gestionados por un microcontrolador LPC1768, ubicado físicamente junto a una de las lámparas LED_i (i = 1), que genera periódicamente las cuatro señales a transmitir. Las secuencias transmitidas son el resultado de la modulación BPSK de los códigos LS c_i de 1151 bits asignados a cada baliza i. A pesar de que todos los códigos transmitidos se mezclan en el receptor, el uso de un código diferente por baliza permite que el algoritmo de posicionamiento, posterior a la recepción, pueda identificar el par código-baliza y extraer así información del ángulo de incidencia, intensidad de la señal, etc., por cada baliza.

La frecuencia portadora de la modulación BPSK es $f_c = 25$kHz, mucho mayor de 200Hz para eliminar el efecto de parpadeo. Las secuencias c_i son transmitidas simultáneamente por cada LED_i con un desplazamiento de 5 muestras con respecto a la anterior, como se muestra en la Figura 9. Esto se hace para mejorar las propiedades de correlación cruzada de los códigos en el receptor.

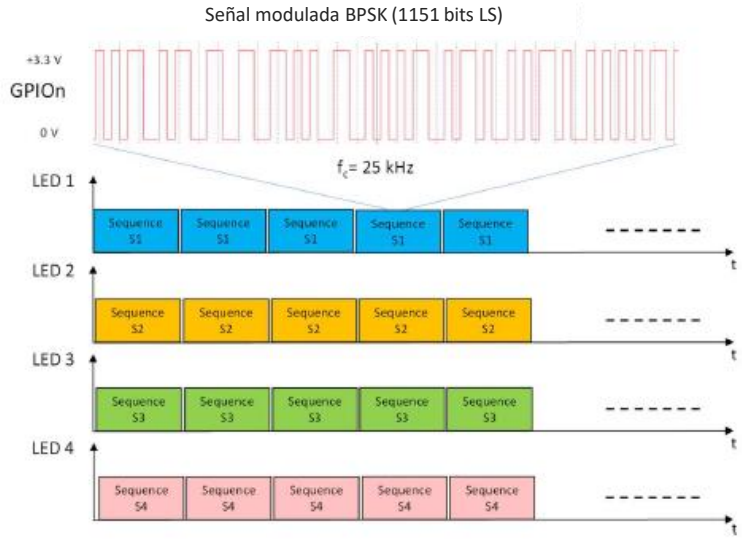

Figura 9. Uno de los códigos c_i generados por LPC1768, ya modulado mediante BPSK.

3.1.2. Receptor

El receptor del sistema es un QADA QP50-6-18u-TO [26] (*Quadrant Photodiode Angular Diversity Aperture*). Está constituido por cuatro fotorreceptores cuyas áreas efectivas individuales son iguales a 11,78 mm^2. Sobre él se instala una abertura cuadrada cuyo lado es igual al diámetro del círculo del fotorreceptor, a una altura h_{ap} por encima de él. La altura y forma de dicha apertura ha sido estudia y analiza en [1] para conseguir el mayor rendimiento del sistema.

A partir de dicho montaje es posible obtener las coordenadas (x_r, y_r) del punto de impacto del rayo incidente en la superficie del receptor QADA, con las cuales puede estimarse geométricamente el ángulo de incidencia Ψ del rayo como se muestra en la Figura 10.

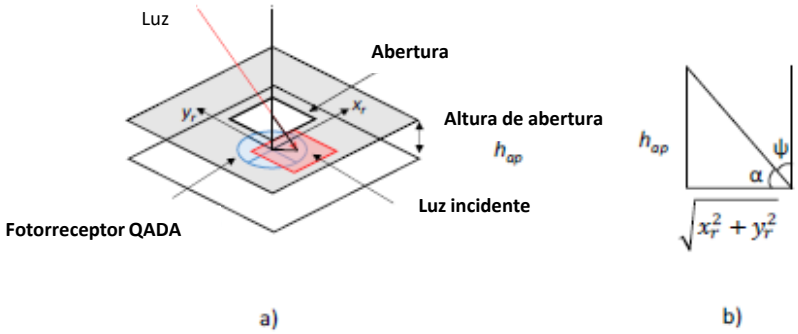

Figura 10. a) Luz incidente en el fotorreceptor QADA; b) Análisis geométrico de la abertura y del punto de impacto [1].

En la *Figura 11* se muestra cómo un haz de luz incide en el fotorreceptor tras pasar por la apertura y cubre las diferentes áreas del mismo. En este ejemplo concreto el haz incide principalmente en el área A1 del QADA.

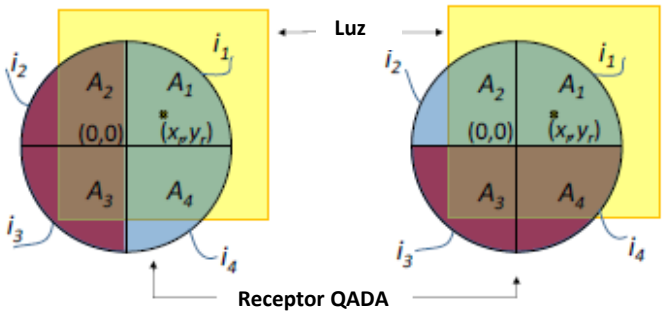

Figura 11. Luz incidente en un receptor QADA a través de una abertura para ambas coordenadas: eje X (izquierda) y eje Y (derecha)[1].

Por otro lado, el módulo QP50-6-18u-TO mostrado en la Figura 12 tiene como salidas la combinación de las corrientes $i_j(t)$ (véase la *Figura 11*) correspondientes a cada área del fotodiodo, transformadas en tensiones $v_j(t)$ mediante un amplificador de transimpedancia [26]. Dichas combinaciones de voltajes entre los cuatro cuadrantes j={1,2,3,4} son: la diferencia de voltaje respecto al eje X, $V_{LR}(t)$; respecto al eje Y, $V_{BT}(t)$; y la suma de todos los cuadrantes, $V_{sum}(t)$. La definición de estas señales se muestra con más detalle en (1)-(3). Cada cuadrante j detectará las contribuciones de todas las secuencias c_i transmitidas por cada LED_i que afecten a la superficie del cuadrante correspondiente.

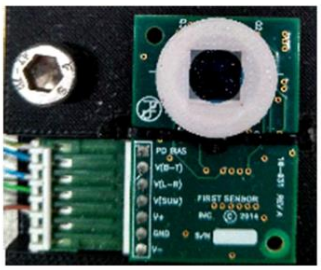

Figura 12. Módulo QP50-6-18u-TO

$$V_{sum}(t) = \sum_{j=1}^{4} v_j(t) \quad (1)$$

$$V_{LR}(t) = \big(v_2(t) + v_3(t)\big) - (v_1(t) + v_4(t)) \quad (2)$$

$$V_{BT}(t) = \big(v_3(t) + v_4(t)\big) - (v_1(t) + v_2(t)) \quad (3)$$

Una vez obtenidas las tres salidas, $V_{LR}(t)$, $V_{BT}(t)$ y $V_{sum}(t)$, son filtradas mediante un circuito inversor paso banda, con el fin de eliminar posibles interferencias de la señal a estudiar con la luz de fondo del interior de la sala donde se verifica el sistema propuesto. Resulta fundamental el uso de dicho filtro, pues como se mostrará a continuación, los errores introducidos por la detección de las señales generarán errores en la triangulación y posicionamiento del receptor.

3.1.3. Algoritmo de posicionamiento

Una vez filtradas las señales, éstas se adquieren y se muestran en un osciloscopio conectado a un ordenador, el cual permite el procesamiento de las señales recibidas para estimar la posición (x_r, y_r, z_r) del receptor. Este proceso, hasta ahora, ha sido realizado íntegramente por el software Matlab ©.

Cabe destacar que el algoritmo de posicionamiento de este sistema está basado en la técnica de posicionamiento de triangulación con ángulo de llegada (AoA). El sistema propuesto estima los ángulos provenientes de las balizas. En la Figura 13 se muestra un esquema explicativo del proceso de estimación de la posición del receptor. Pueden diferenciarse los siguientes pasos:

- Correlación de las señales detectadas por el receptor QADA con las cuatro secuencias LS c_i.
- Búsqueda de picos máximos de las doce correlaciones realizadas.
- Cálculo de las relaciones p_x y p_y.
- Cálculo del punto de incidencia del haz de luz en el receptor QADA.
- Cálculo de la posición estimada del receptor.

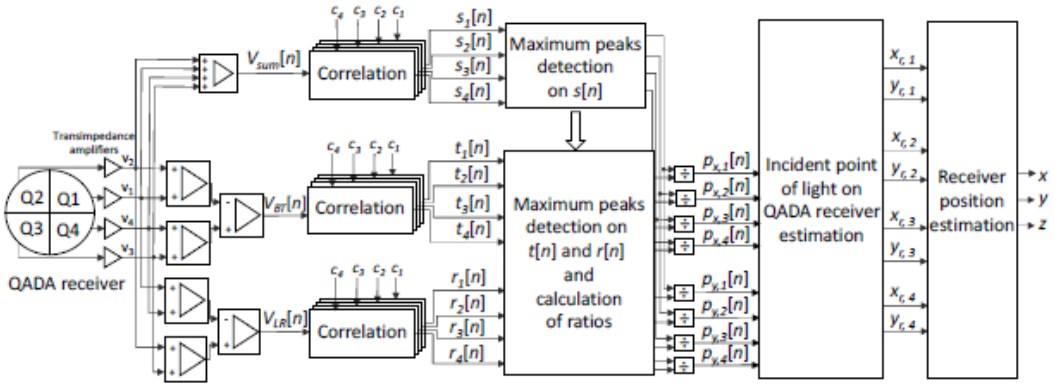

Figura 13. Procedimiento global para la estimación de la posición del receptor [1].

3.2. Propuesta de representación en coma fija del algoritmo para la determinación del punto de impacto

Como se ha mencionado anteriormente, el objetivo de este TFG es el diseño de una arquitectura en el lenguaje VHDL que reproduzca el procesamiento realizado por el software Matlab. Es decir, se quiere implementar el algoritmo de posicionamiento propuesto en hardware. Por lo tanto, es necesario estudiar la representación de las variables del código, que están almacenadas en coma flotante, en coma fija, teniendo como objetivo encontrar el formato idóneo que permita representar las variables con la mayor precisión, cometiendo así el menor error posible. Es necesaria dicha conversión, de coma flotante a coma fija, porque los sistemas HW, las FPGAS, trabajan con datos en coma fija.

En los siguientes puntos se procede a analizar paso a paso el algoritmo de posicionamiento desarrollado en Matlab en [1]. En el presente documento no se ha tratado todo el algoritmo representado en la Figura 13, sino que, como se muestra en la Figura 14, la parte del algoritmo

a implementar en HW se centra en las fases necesarias para determinar el punto de impacto del haz de luz sobre el receptor QADA para cada emisor LED.

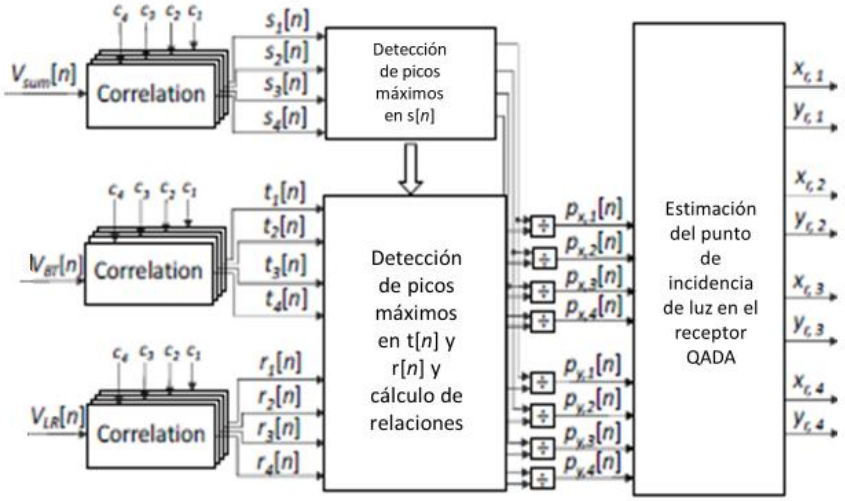

Figura 14. Esquema simplificado de los cálculos abordados en este TFG.

Para realizar la adecuación de los diferentes procesos se ha hecho uso del software Matlab. Se ha estudiado la conversión de las variables del formato en coma flotante al formato de coma fija Q*n.m*, buscando para cada variable el número total de bits necesarios (*n*+*m*) y especificando cuántos de ellos representarán la parte entera, *n*, y cuántos la parte fraccionaria, *m*. Para ello se han utilizado una serie de funciones de Matlab:

- **Quantizer**: crea un objeto cuantificador con las propiedades que se indiquen en su argumento. Pudiendo especificar:
    - El **tipo de aritmética** utilizada en la cuantificación: cálculos de punto fijo con signo, '*fixed*', de punto flotante, '*float*', de punto flotante de doble precisión, '*double*', de punto flotante de precisión simple, '*single*' o de punto fijo sin signo, '*ufixed*'. Si no se le indica lo contrario, por defecto, Matlab fija la propiedad '*fixed*'.
    - El **formato de datos** del objeto cuantificador. La interpretación de este valor dependerá del tipo de aritmética seleccionada. Si se ha fijado un modo de punto fijo para números con o sin signo el formato tendrá el siguiente patrón: [*wordlength fractionlength*]. Mientras que si se ha fijado una aritmética de punto flotante el formato será: [*wordlength exponentlength*].
    - La forma de manejar un **desbordamiento**. Pudiendo elegir entre: '*saturate*', si conviene que la variable se fije en el valor máximo o mínimo representable en función de cuál sea el más cercano. O bien '*Wrap*', donde el software no realiza una extensión de signo cuando el resultado de una operación desborda.
    - El método de **redondeo**. '*Ceiling*' redondea al siguiente valor cuantificado permitido. '*Convergent*', que redondea al valor cuantificado permisible más cercano. 'Zero', redondea los números negativos hacia arriba y los números positivos hacia abajo hasta el siguiente valor cuantificado permitido. '*Floor*', valor predeterminado si no se indica lo contrario, redondea al siguiente valor

cuantificado permitido. Por último, '*Nearest*', redondea al valor cuantificado permisible más cercano.

- **Quantize:** cuantifica números de coma fija, pasándole como argumentos el objeto cuantificador *quantizer* y el valor a cuantificar.

Por lo tanto, para adecuar un valor, en primer lugar se crea un objeto cuantificador. Se le indica el tipo de aritmética a utilizar, cómo se quiere que el valor sea redondeado, en el caso de que el número de bits no permita representar el número exacto, qué hacer si el número satura y, por último, el formato de los datos. Después se efectúa la cuantificación con *quantize*. En los siguientes puntos se detalla el formato Q*n.m* elegido para cada variable del algoritmo y los errores de cuantificación cometidos para cada una de ellas.

3.2.1. Representación en coma fija del algoritmo de correlación

El objetivo de la correlación es poder identificar, en cada señal aportada por el QADA, cuáles de las secuencias LS c_i han sido captadas con más o menos intensidad, permitiendo así estimar la posición del punto de incidencia (x_r, y_r) de cada lámpara LED.

Las señales obtenidas a la salida del receptor QADA son correlacionadas con cada uno de los cuatro códigos emitidos por los LEDs (véase la Figura 13), siendo por tanto $s_i[n]$, $t_i[n]$ y $r_i[n]$, el resultado de las correlaciones entre las señales recibidas $V_{LR}(t)$, $V_{BT}(t)$ y $V_{sum}(t)$, respectivamente, y la secuencia LS c_i correspondiente para cada LED_i, las cuales son conocidas por el receptor.

Para iniciar el proceso de cuantificación del algoritmo de posicionamiento, se ha empezado por adecuar las señales de entrada a la arquitectura a diseñar o, lo que es lo mismo, las señales de salida del receptor QADA, es decir, $V_{LR}(t)$, $V_{BT}(t)$ y $V_{sum}(t)$. Puesto que dichas señales habrán de ser adquiridas por el ADC de la FPGA, se asume de partida que éste trabaja con una longitud de palabra de 12 bits, codificándose así con 12 bits. Cabe destacar que, en lo que resta de documento, las señales $V_{LR}(t)$, $V_{BT}(t)$ y $V_{sum}(t)$ serán referidas como *codevlr*, *codevbt* y *codevsum* respectivamente.

Se representaron gráficamente las señales para hacer un estudio previo del rango de valores de cada una de ellas y poder estimar así cuántos bits adjudicar a la parte entera y cuántos a la parte fraccionaria. Observando la Figura 15 se llegó a la conclusión de que no era necesario ni siquiera un bit en la parte entera, pues los valores de las señales son del orden de 10^{-4}, siendo el mayor igual a 0.0012.

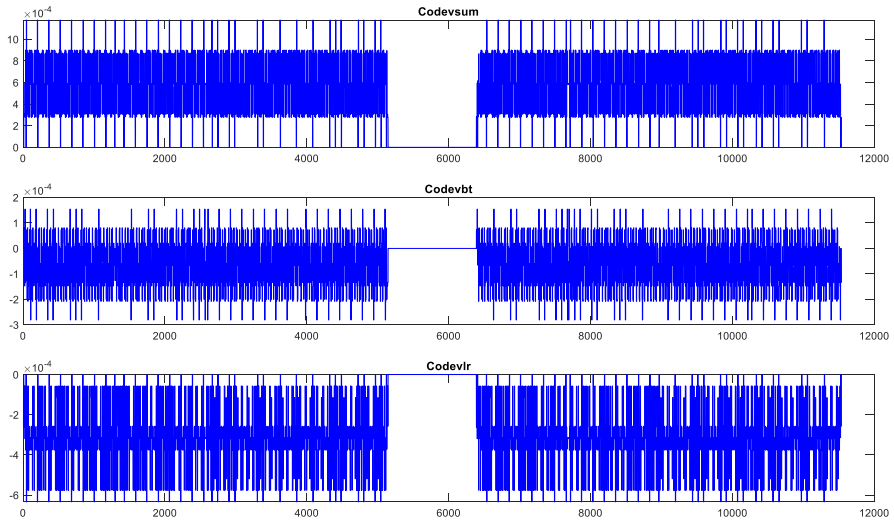

Figura 15. Representación gráfica de las señales de salida del receptor QADA.

Por ello, en un primer momento se fija un formato Q0.12, obteniéndose el resultado de la Figura 16. Se ha representado solo una de las señales pues los valores de las tres son muy similares y haciendo el estudio para una de ellas puede extrapolarse el resultado para las demás. Este razonamiento se extiende a lo largo de todo este apartado.

En la primera fila de la Figura 16, se muestra la señal *codevbt* real en azul y, superpuesta en rojo, la representada en coma fija. En la segunda fila puede observarse el error de cuantificación cometido. Es evidente que se trata de un error significativo, pues es del orden de las señales reales. Por lo tanto, se aumenta el peso de los bits de la parte fraccionaria para conseguir así una mayor resolución.

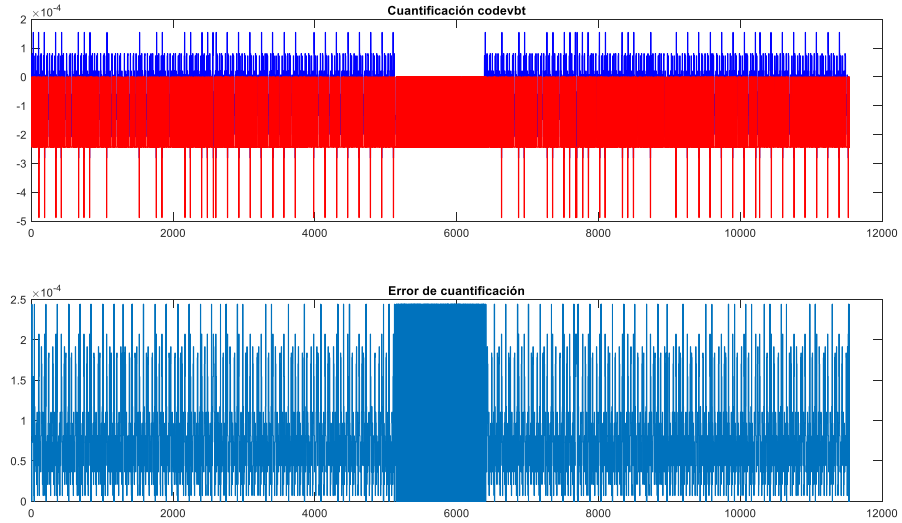

Figura 16. Cuantificación de la señal codevbt con un formato Q0.12 y error de cuantificación.

Se realizan simulaciones para un formato Q-2.12, donde el error mejora en un orden de magnitud, pero sigue siendo algo significativo como puede observarse en la Figura 17. Finalmente se fija el formato Q-6.12, que significa que, de los 12 bits máximos que puede tener el dato, 18 son fraccionarios. Esto hace que el peso de los bits fraccionarios aumente, consiguiendo de esta manera una precisión suficiente para que el error cometido sea

prácticamente despreciable. En la Figura 18, Figura 19 y Figura 20 se muestran los resultados de las cuantificaciones junto con los errores. Ahora, dicho error es del orden de 10^{-6}, lo cual se considera aceptable por ser varios órdenes de magnitud inferior a la amplitud de las señales originales. En la Tabla 1, se han reflejado todos los errores máximos relativos para cada formato de coma fija simulado, resaltándose el formato elegido.

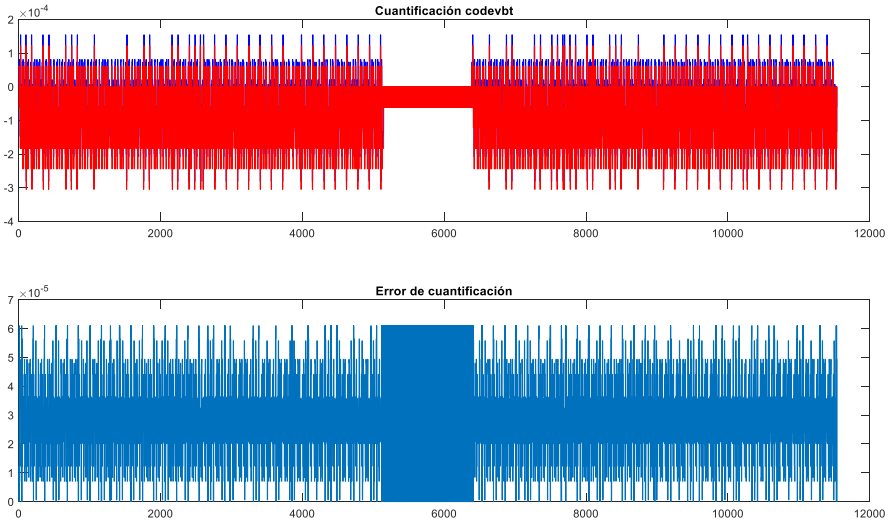

Figura 17. Cuantificación de la señal codevbt con un formato Q-2.12 y error de cuantificación.

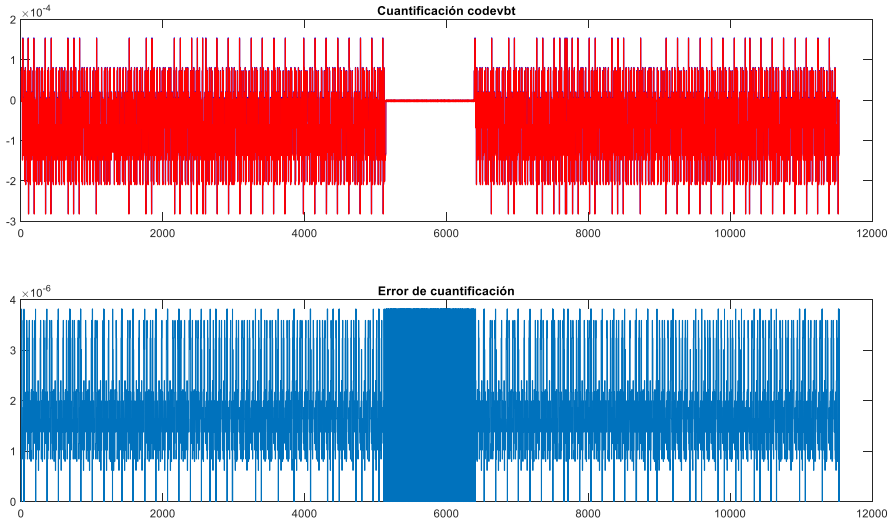

Figura 18. Señal codevbt real, cuantificada y error de cuantificación Q-6.12.

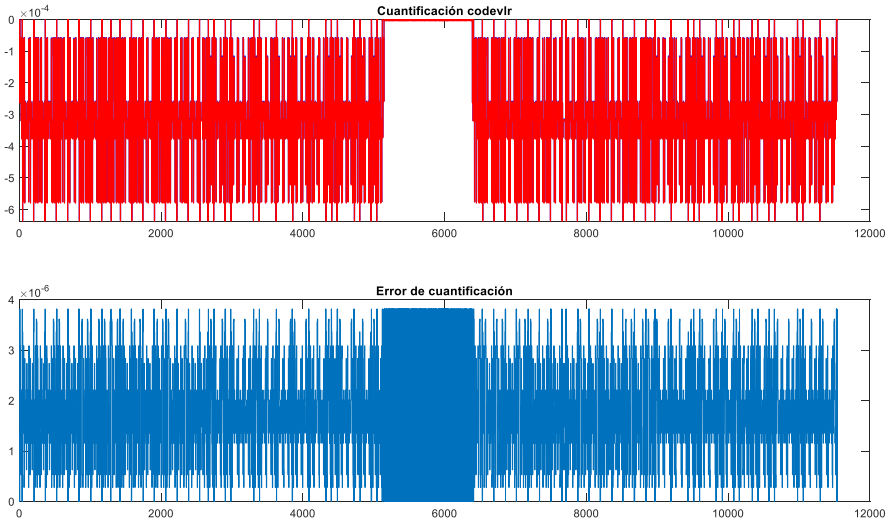

Figura 19. Señal codevlr real, cuantificada y error de cuantificación Q-6.12.

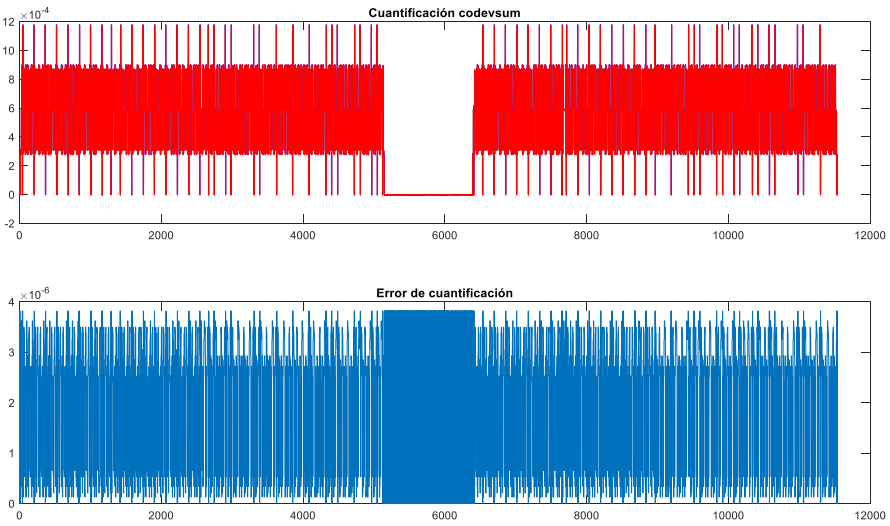

Figura 20. Señal codevsum real, cuantificada y error de cuantificación Q-6.12.

Para realizar estas simulaciones se ha hecho uso de las funciones mencionadas anteriormente. En la Figura 21, se muestra la porción de código donde se llama a dichas funciones. En este caso se ha fijado el modo de desbordamiento como 'saturate'. Éste suele utilizarse para cuantificar señales externas debido a que, por lo general, un ADC al cuantificar satura si no tiene bits suficientes para representar un valor. Por lo tanto, es una forma de realizar una simulación realista.

```
NBits_Input= [12 12+6]; %REPRESENTACIÓN ELEGIDA
%NBits_Input = [12 12];
%NBits_Input = [12 12+2];

Q= quantizer('fixed', 'floor', 'saturate', NBits_Input);
codevsumq= quantize (Q, codevsum);
codevlrq= quantize (Q, codevlr);
codevbtq= quantize (Q, codevbt);
```

Figura 21. Sección de código de Matlab donde se realiza la cuantificación de las señales de salida del QADA.

Formato de Representación	Error Relativo del canal de suma	Error relativo del canal LeftRight	Error relativo del canal BottonTop
Q0.12	20.7254%	38.5267%	86.8407%
Q-2.12	5.1813%	9.6317%	21.7102%
Q-6.12	**0.3238 %**	**0.6020%**	**1.3569%**

Tabla 1. Errores relativos de la señal de entrada para cada formato de representación.

Una vez adecuadas las señales de salida del receptor QADA, se procede a estudiar los códigos LS c_i. Cabe recordad que éstos han de ser conocidos por el sistema de procesamiento para poder realizar las correlaciones. De ahora en adelante los códigos son renombrados como *sMod*.

Se ha seguido el mismo procedimiento que con las señales anteriores. En este caso los patrones varían entre 1 y -1, como puede verse en la Figura 22, por ello se elige una configuración Q1.11. De esta manera podrá obtenerse un -1 exacto pero no se llegará al 1, obteniéndose en su defecto un 0.9995. Se observa que el error cometido es prácticamente despreciable. En la Tabla 2 se muestran los errores máximos relativos de cada uno de los códigos emitidos.

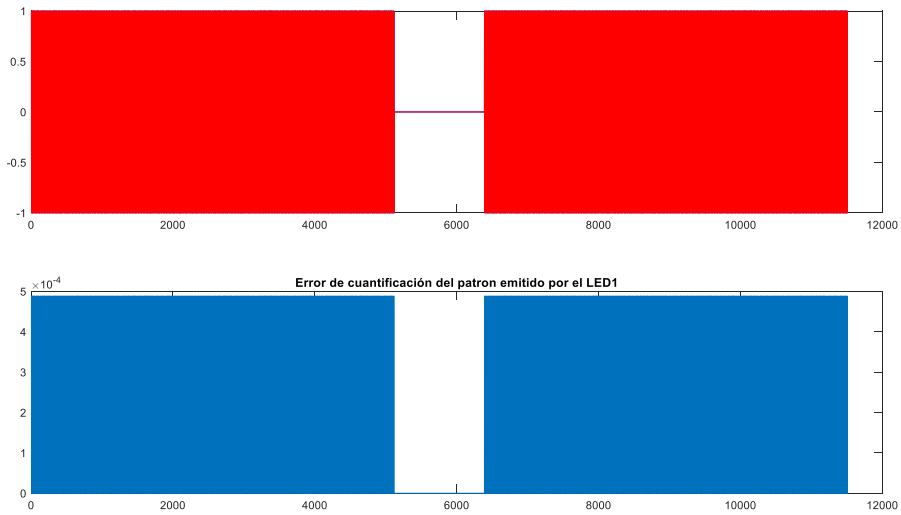

Figura 22. Cuantificación del código del LED_1 junto con el error de cuantificación.

A continuación, en la Figura 23, se muestra la sección de código que realiza la cuantificación de los patrones. De nuevo se ha utilizado el modo de desbordamiento *'saturate'*, por lo razonado anteriormente.

```
%Cuantificacion del patron
NBits_patron= [12 11];
Q_patron= quantizer('fixed', 'floor', 'saturate', NBits_patron);
sModq= quantize (Q_patron, sMod);
```

Figura 23. Sección de código correspondiente a la cuantificación de las secuencias LS c_i

Formato de representación	Error relativo del patrón LED1	Error relativo del patrón LED2	Error relativo del patrón LED3	Error relativo del patrón LED4
Q1.11	**0.0488%**	**0.0488%**	**0.0488%**	**0.0488%**

Tabla 2. Errores relativos de las balizas emitidas por cada LED para el formato de representación fijado.

Finalmente, se procede a la adecuación de la correlación. Observando la función que realiza dicha operación en el código de Matlab se observa que, entre otras operaciones, hay una suma y una multiplicación. Sabiendo que dicha multiplicación va a realizarse entre una de las señales

de salida del QADA y un código, con formato en coma fija Q-6.12 y Q1.11 respectivamente, el resultado necesitará tener un formato Q-5.24 para poder ser representado con una precisión aceptable. Para verificar este razonamiento se ha simulado la multiplicación real y estimada con dicho formato, obteniéndose el resultado de la Figura 24. Se muestra solo la interpretación de la multiplicación entre valores de la señal *codevbt* y del código emitido por el LED 2, por ser el resto de representaciones muy similares a ésta. Dentro de la similitud, estas señales son más representativas por presentar errores mayores, como se muestra en Tabla 3, Tabla 4 y Tabla 5. Dichos errores se han considerados aceptables.

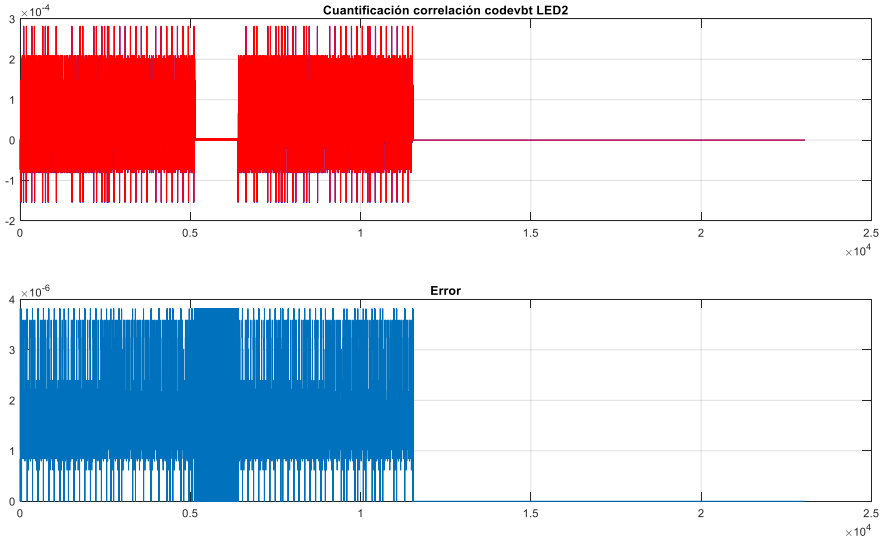

Figura 24. Cuantificación de la multiplicación y su error para el LED2.

Formato de representación	Error máximo relativo LED1	Error máximo relativo LED2	Error máximo relativo LED3	Error máximo relativo LED4
Q-5.24	**0.3319%**	**0.3238%**	**0.3238%**	**0.3329%**

Tabla 3. Errores máximos relativos cometidos en la representación en coma fija de la multiplicación entre la señal del canal suma y los códigos emitidos por cada LED_i.

Formato de representación	Error máximo relativo LED1	Error máximo relativo LED2	Error máximo relativo LED3	Error máximo relativo LED4
Q-5.24	**0.6017%**	**0.6020%**	**0.6020%**	**0.6017%**

Tabla 4. Errores máximos relativos cometidos en la representación en coma fija de la multiplicación entre la señal del canal LR y los códigos emitidos por cada LED_i.

Formato de representación	Error máximo relativo LED1	Error máximo relativo LED2	Error máximo relativo LED3	Error máximo relativo LED4
Q-5.24	**1.3569%**	**1.3539%**	**0.3539%**	**0.3539%**

Tabla 5. Errores máximos relativos cometidos en la representación en coma fija de la multiplicación entre la señal del canal BT y los códigos emitidos por cada LED_i.

Tras la multiplicación, se van acumulando los valores de los sucesivos resultados, como se muestra en la Figura 25. En ella se resalta la operación de multiplicación y acumulación, donde la variable *output2* contiene el resultado de la muestra de correlación *j* del LED_i. Ha de estudiarse el posible desbordamiento producido por esta acumulación.

```
for i = 1:noLEDs
    % output(i,:) = conv(signal, flip(sMod(i,:)));
    % Algoritmo propio
    %sMod_ext= flip(sMod(i,:));
    sMod_ext= sMod(i,:);
    for j=1:length(signal_ext)-length(sMod_ext)
        output2(i,j)= 0;
        for m=1:length(sMod_ext)
            output2(i,j) = output2(i,j) + signal_ext(j+m)*sMod_ext(m);
        end;
    end;

    subplot(noLEDs+1,1,i+1);
    plot(output2(i,:));

    output2 = quantize(Q_prod, output2);
end
```

Figura 25. Algoritmo de correlación.

Fijando de nuevo la atención en la Figura 25, se observa que la operación de multiplicación y acumulación, resaltada en rojo, se realiza tantas veces como columnas tiene la variable *sMod_ext*. Esta variable almacena las 11510 muestras de cada uno de los cuatro códigos emitidos por cada LED. Las balizas emitidas por los LEDs son códigos LS de 1151 bits modulados con BPSK y con una longitud de símbolo de 10 muestras para un ciclo de portadora, lo cual constituye las 11510 muestras transmitidas por cada LED. Suponiendo que en cada iteración del bucle, como mínimo, se aumente la variable *output2* en una unidad, esta podrá llegar a un valor máximo igual a 11510. Se sabe que para representar dicha cifra son necesarios al menos 14 bits ($\log_2(11510) = 13.49 \approx 14\ bits$), por lo tanto, se puede suponer también que *output2* debe tener al menos 14 bits más de los razonados para la multiplicación, para poder así representar dicha acumulación con suficiente precisión. Por lo tanto, al acumular un dígito con formato Q-5.24, correspondiente al resultado de la multiplicación, con otro con formato Q14.0, se obtiene un formato Q9.29, es decir, 38 bits de los cuales 29 conforman la parte fraccionaria y 9 la entera. En aras de poder adecuar el resultado a un bus estándar ofrecido por la FPGA, la longitud total de la variable final que contiene el resultado de las correlaciones se trunca a 32 bits.

Tras este razonamiento se realizaron una serie de pruebas para determinar qué formato es el más adecuado:

- **Q16.16:**

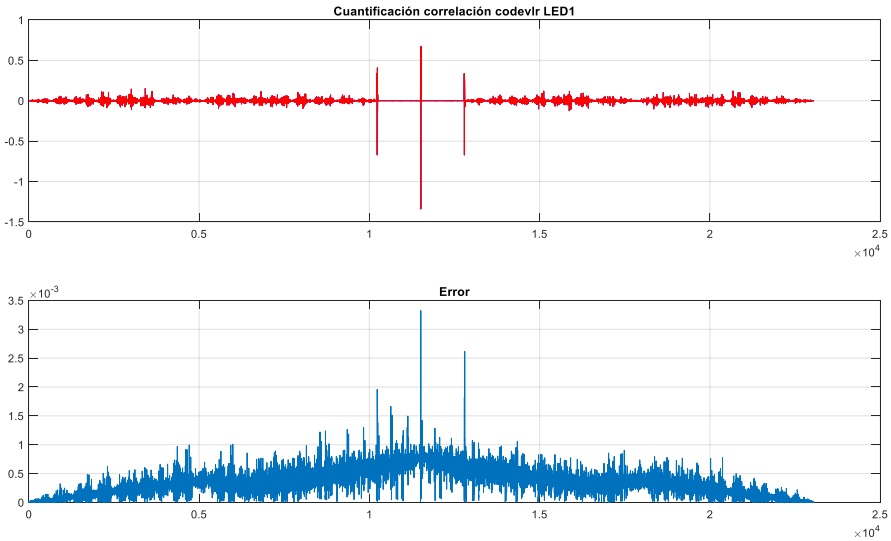

Figura 26. Representación de la señal del canal LR real y en coma fija con formato Q16.16, junto con el error cometido.

Con el formato de representación en coma fija Q16.16, se tiene un error más o menos aceptable, del orden de 10^{-3}, pero se estudian otras configuraciones.

- **Q5.27:**

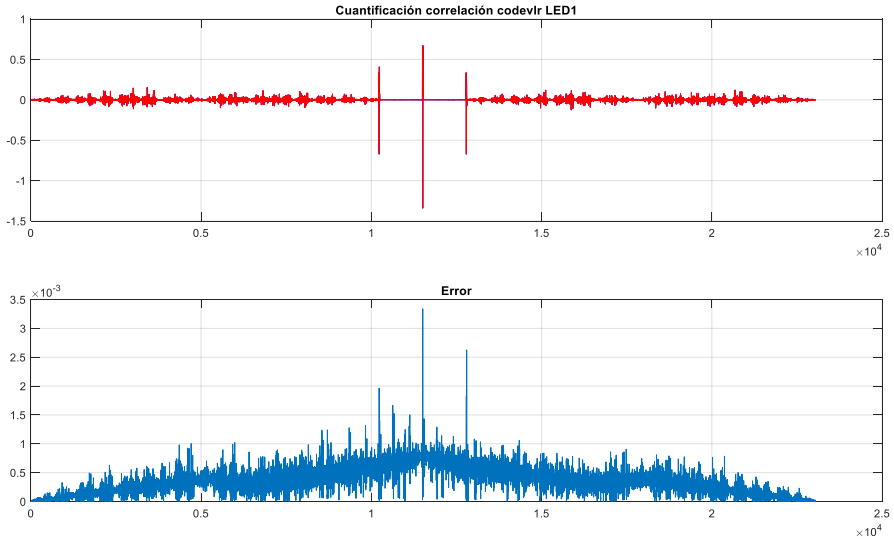

Figura 27. Representación de la señal del canal LR real y en coma fija con formato Q5.27, junto con el error cometido.

- **Q3.29:**

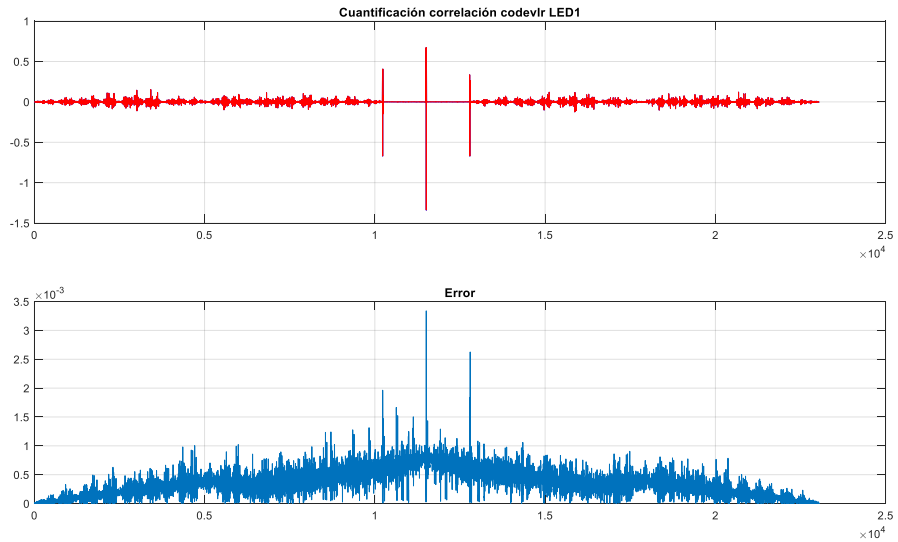

Figura 28. Representación de la señal del canal LR real y en coma fija con formato Q3.29, junto con el error cometido.

- **Q2.30:**

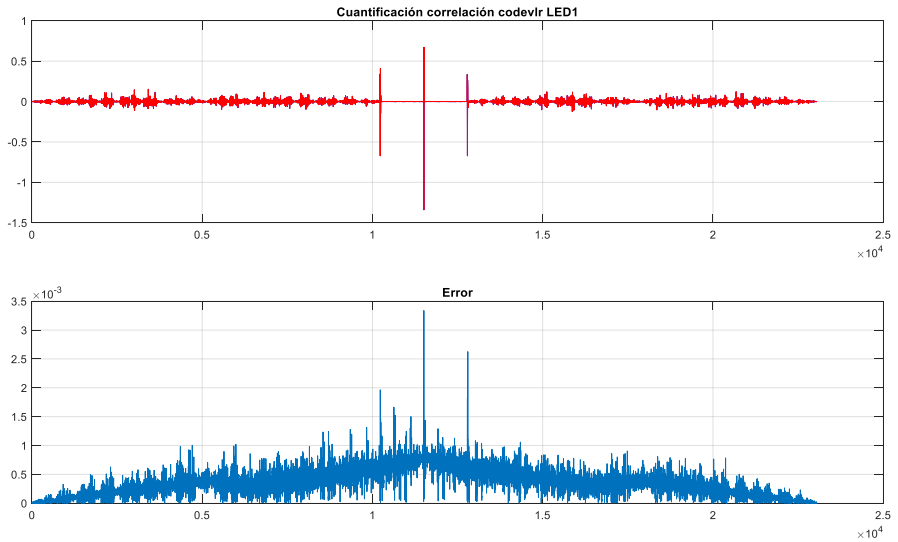

Figura 29. Representación de la señal del canal LR real y en coma fija con formato Q2.30, junto con el error cometido.

- **Q1.31:**

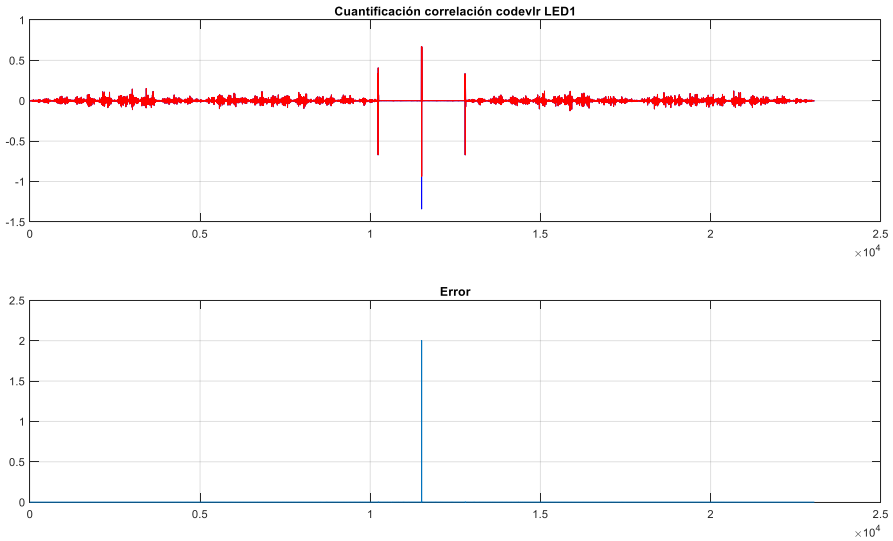

Figura 30. Representación de la señal del canal LR real y en coma fija con formato Q1.31, junto con el error cometido.

Por último, en la Tabla 6, Tabla 7 y Tabla 8 se muestran los errores máximos relativos de las correlaciones de las señales de salida del QADA con cada uno de los códigos emitidos por los LEDs. Tras analizar los resultados y observar que con algunos formatos se obtienen unas representaciones muy similares, se ha elegido el formato Q3.29, por mantener el número de bits de la parte fraccionaria razonada anteriormente de forma teórica.

Formato de representación	Error máximo relativo LED1	Error máximo relativo LED2	Error máximo relativo LED3	Error máximo relativo LED4
Q16.16	0.1894%	0.2801%	0.3521%	0.7183%
Q5.27	0.1893%	0.2799%	0.3513%	0.7177%
Q3.29	**0.1893%**	**0.2799%**	**0.3513%**	**0.7177%**
Q2.30	0.1893%	0.2799%	0.3513%	0.7177%
Q1.31	138.3255%	125.2857%	127.7633%	141.5455%

Tabla 6. Errores máximos relativos de representación en coma fija del resultado de las correlaciones de la señal del canal suma con cada uno de los códigos emitidos por cada LED.

Formato de representación	Error máximo relativo LED1	Error máximo relativo LED2	Error máximo relativo LED3	Error máximo relativo LED4
Q16.16	0.2479%	0.4961%	0.7199%	0.1981%
Q5.27	0.2490%	0.4947%	0.7220%	0.1981%
Q3.29	**0.2490%**	**0.4947%**	**0.7220%**	**0.1981%**
Q2.30	0.2490%	0.4947%	0.7220%	0.1981 z%
Q1.31	149.421 %	0.4947%	0.7220%	151.8006%

Tabla 7. Errores máximos relativos de representación en coma fija del resultado de las correlaciones de la señal del canal LR con cada uno de los códigos emitidos por cada LED.

Formato de representación	Error máximo relativo LED1	Error máximo relativo LED2	Error máximo relativo LED3	Error máximo relativo LED4
Q16.16	0.1748%	0.2355%	0.3697%	0.2775%
Q5.27	0.1758%	0.2355%	0.3687%	0.2758%
Q3.29	**0.1758%**	**0.2355%**	**0.3687%**	**0.2758%**
Q2.30	0.1758%	0.2355%	0.3687%	0.2758%
Q1.31	0.1758%	0.2355%	0.3687%	0.2758%

Tabla 8. Errores máximos relativos de representación en coma fija del resultado de las correlaciones de la señal del canal BT con cada uno de los códigos emitidos por cada LED.

Para realizar la adecuación de las correlaciones a formato de coma fija se han generado líneas de código muy similares a las mostradas en la Figura 21 pero con una salvedad. En este caso el modo de desbordamiento elegido ha sido '*wrap*', como puede observarse en la Figura 31. A partir de este momento, todas las variables restantes se convertirán con este modo de desbordamiento, ya que simula el comportamiento interno real de una FPGA. Cuando una variable desborda, no se realiza una extensión de signo para dar el resultado correcto, pudiendo obtenerse entonces, por ejemplo, un resultado negativo de una suma aritmética de dos números positivos.

```
NBits_prod= [32 29];%ELEGIDO
Q_prod= quantizer('fixed', 'floor', 'wrap', NBits_prod);
....
....
output2 = quantize(Q_prod, output2);
```

Figura 31. Sección de código de Matlab donde se genera el objeto cuantificador y la adecuación del resultado de las correlaciones.

3.2.2. Representación en coma fija de las relaciones $\boldsymbol{p_x}$ y $\boldsymbol{p_y}$

Tras la realización de las doce correlaciones, el algoritmo busca los valores máximos positivos o negativos de cada correlación. Aquellos LEDs ubicados a mayores distancias del receptor ofrecen peores funciones de correlación, lo que puede traducirse en una mayor dificultad de detección de máximos. Por ello, el algoritmo propuesto en [1] localiza primero los picos máximos en las funciones de correlación del canal de suma. Estas correlaciones siempre presentan amplitudes mayores respecto al resto, lo cual favorece la detección más precisa de máximos. Después, se localizan los demás picos para las correlaciones de los canales restantes teniendo en cuenta los desplazamientos de las transmisiones realizadas. Es crucial llevar a cabo una correcta detección de dichos picos de correlación para poder obtener la posición del receptor con un error de posicionamiento mínimo.

Para dicho algoritmo de búsqueda de máximos no se ha realizado ningún estudio de representación en coma fija pues no se ejecuta ninguna operación aritmética que genere nuevas variables a estudiar o que pueda desbordar variables ya estudiadas.

Tras hallarse los máximos de las correlaciones para cada LED, se realiza el cálculo de las relaciones px y py, definidas teóricamente en [1] como:

$$p_{x,i}[n] = \frac{\max(r_i[n])}{\max(s_i[n])} \quad (4)$$

$$p_{y,i}[n] = \frac{\max(t_i[n])}{\max(s_i[n])} \quad (5)$$

Siendo $r_i[n]$, $t_i[n]$ y $s_i[n]$ las funciones de correlación de los canales VLR, VBT y de suma respectivamente.

Los parámetros p_x y p_y se obtienen de dividir variables con formato Q3.29, precisión de representación en coma fija elegida para la salida de las correlaciones. Se sabe que las funciones de correlación del canal suma tienen una mayor amplitud que el resto de correlaciones y por ello, el resultado de las ecuaciones (4) y (5) siempre será menor que el valor del numerador, no siendo necesario por tanto aumentar el número de bits para su representación. Se mantiene entonces el formato en coma fija Q3.29, obteniéndose los resultados reflejados en la Figura 32 para p_x y en la Figura 33 para p_y para una captura de ejemplo. En estas dos figuras y en las sucesivas representaciones mostradas de las relaciones p_x y p_y, el eje *x* hace referencia al LED al que le corresponde el valor de la relación.

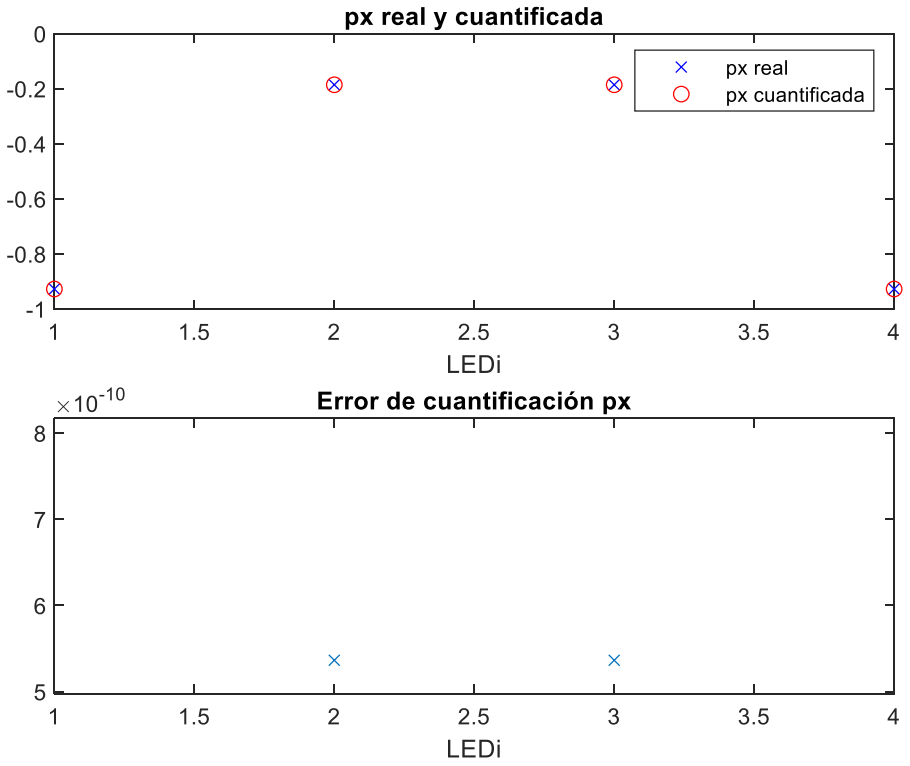

Figura 32. Representación de px con formato Q3.29.

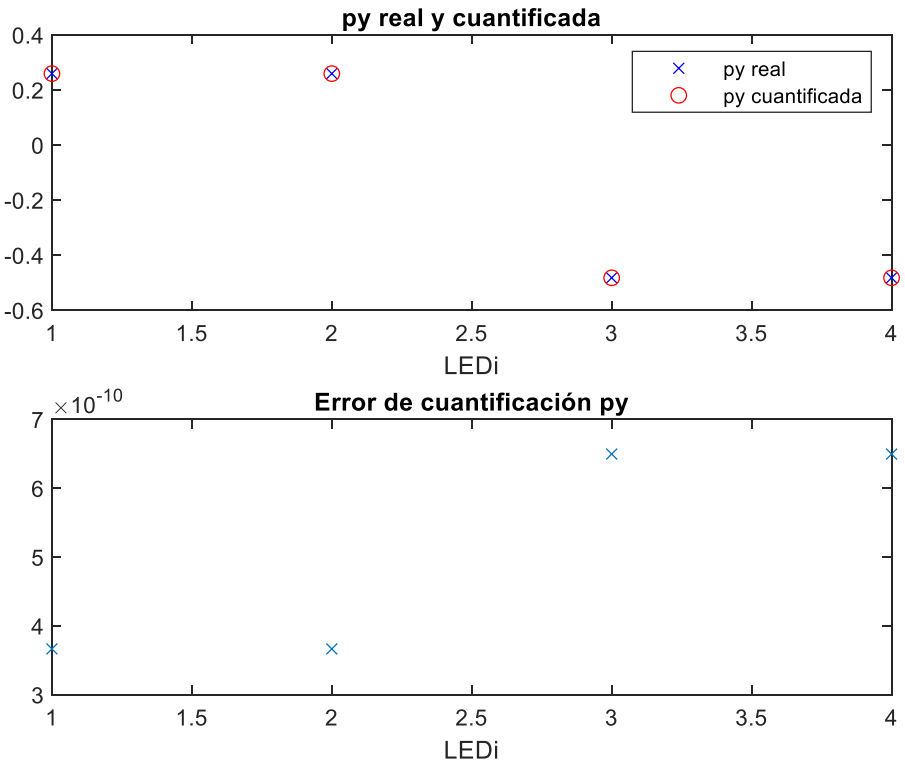

Figura 33. Representación de py con formato Q3.29.

Tras observar que el error cometido es muy pequeño, del orden de 10^{-10}, se ha estudiado la posibilidad de reducir el número de bits necesarios para la representación de estas variables.

Se han analizado las siguientes posibilidades:

- **Reducir a 24 bits con una precisión de Q3.21**

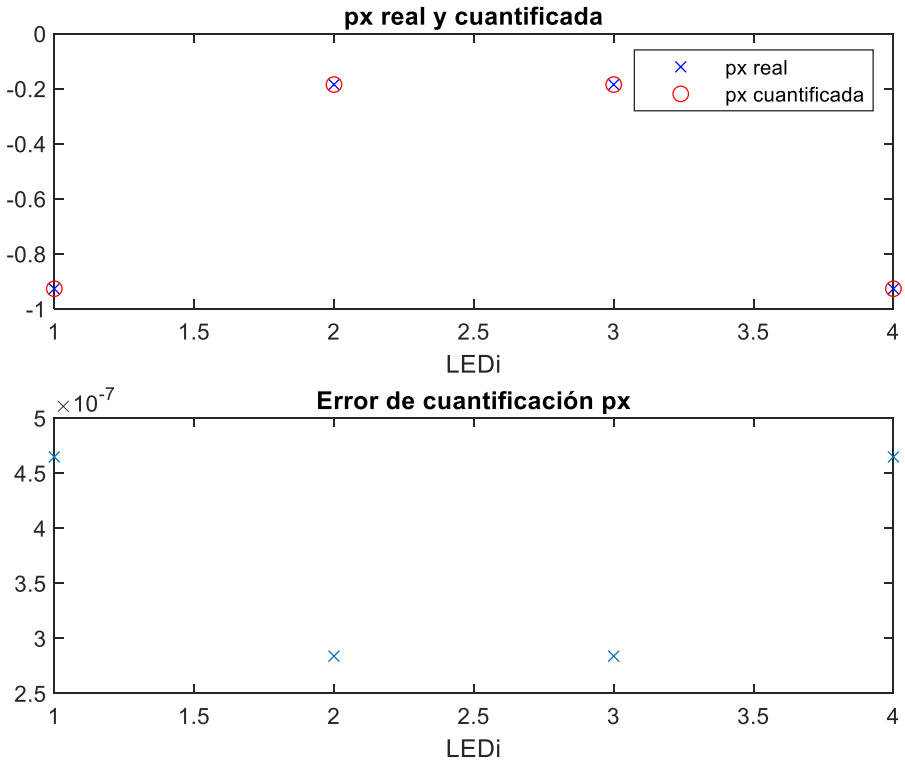

Figura 34. Representación de px con formato Q3.21.

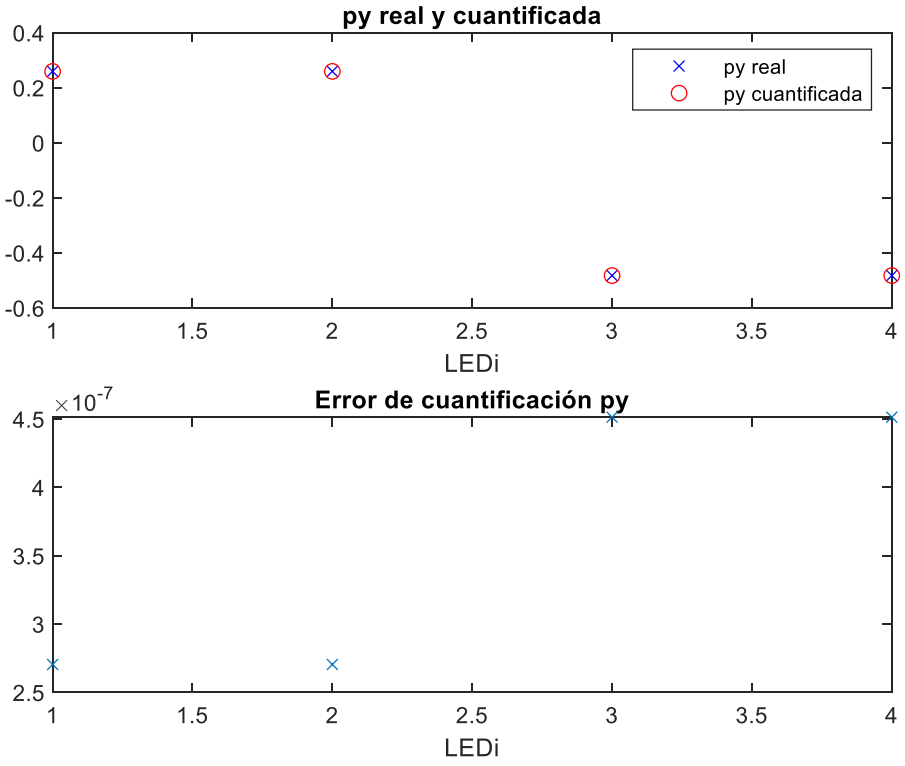

Figura 35. Representación de py con formato Q3.21.

- **Reducir a 16 bits con una precisión de Q3.13**

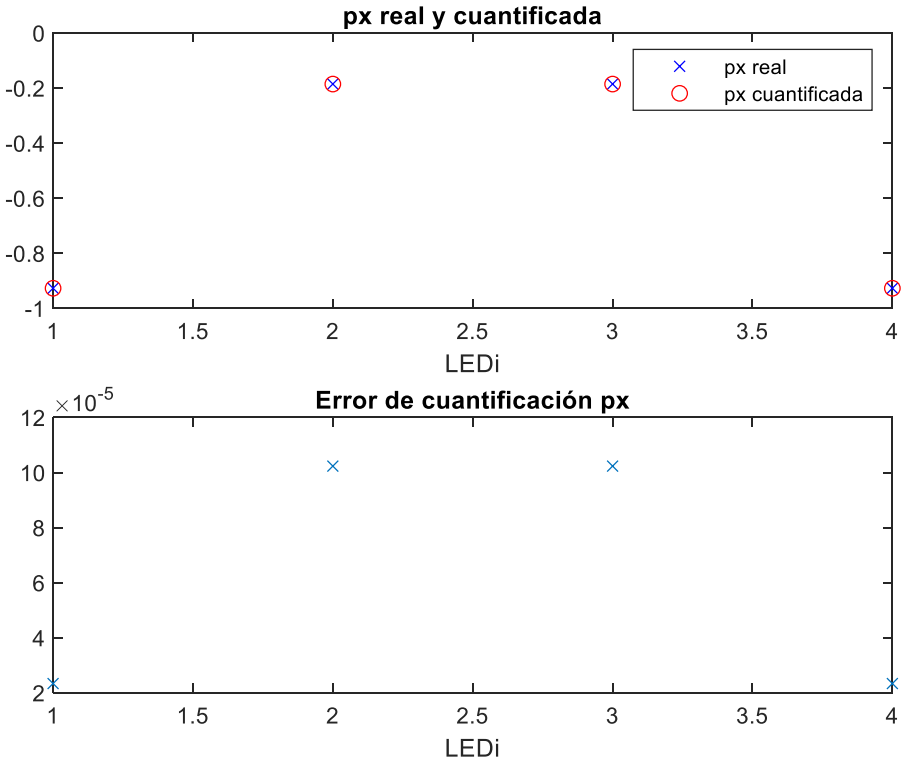

Figura 36. Representación en coma fija de px con formato Q3.13.

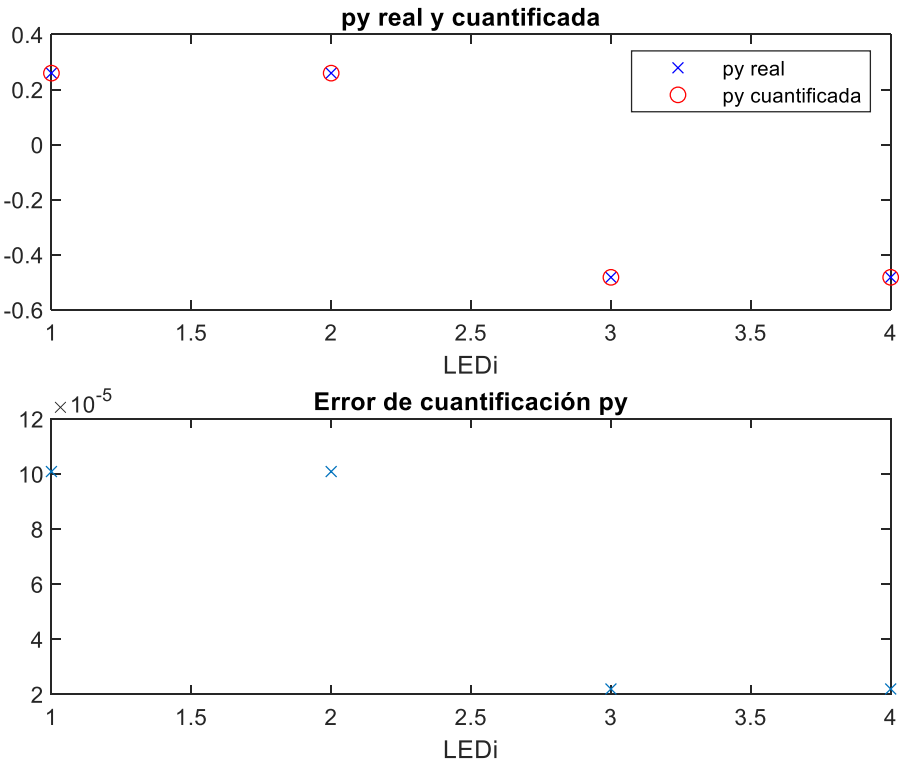

Figura 37. Representación en coma fija de py con formato Q3.13

En la Tabla 9, se muestra un resumen de los errores máximos relativos cometidos para los distintos formatos de representación en coma fija estudiados. Se observa que las tres opciones presentan una buena precisión, por lo que finalmente se elige el formato de 16 bits, pudiendo reducir así los recursos utilizados y consiguiéndose un diseño más eficiente.

Formato de representación	Error relativo px	Error relativo py
Q3.29	8.8382E-8 %	1.3475E-7%
Q3.21	5.0109E-5%	9.3623E-5%
Q3.13	**0.011%**	**0.0209%**

Tabla 9. Errores máximos relativos de la representación de px y py para diferentes formatos.

3.2.3. Adecuación del cálculo del punto de impacto

A partir de las relaciones p_x y p_y, se calculan los puntos de impacto del rayo incidente en la superficie del receptor QADA para cada transmisor LED. Para ello el código de Matlab realiza la operación mostrada en la Figura 38.

$$\text{sol} = [\underbrace{-lambda \cdot l/2 \cdot px \; - \; l/2 \cdot \; lambda \; \cdot \; delta \; \cdot \; py \; + \; xc}_{coordenada\ x}\,;$$
$$\underbrace{lambda \cdot delta \cdot l/2 \cdot lamda \cdot py + yc}_{coordenada\ y}\,]$$

Figura 38. Cálculo de las coordenadas de los puntos de impacto para cada LED.

Fijando a modo demostrativo en lo sucesivo que *lambda* = 1, *l* = 2.75, *delta* = 0, *xc* = 0 e *yc* = 0, siendo todas constantes, se puede realizar la simplificación y el estudio previo mostrado en la Figura 39 para las distintas representaciones en coma fija manejadas.

$$sol = [\overbrace{-lambda}^{Q1.0} * \overbrace{l/2}^{Q2.3} * \overbrace{px}^{Q3.13} ; \quad \overbrace{-l/2}^{Q2.3} * \overbrace{lambda}^{Q1.0} * \overbrace{py}^{Q3.13}]$$
$$Q1.0 + Q2.3 + Q3.13 = Q6.16$$
$$Q2.3 + Q1.0 + Q3.13 = Q6.16$$

Figura 39. Simplificación y estudio de formato en coma fija de los puntos de impacto.

En principio se puede concluir que para representar la posición del punto de impacto son necesarios aproximadamente 22 bits. A partir de esta primera aproximación se han realizado una serie de simulaciones para estudiar el error generado y ver si, como en el caso de las relaciones p_x y p_y [44], se puede reducir el número de bits necesarios manteniendo un error aceptable.

Se han estudiado los siguientes formatos de representación:

- **Con 24 bits, formato Q2.22**

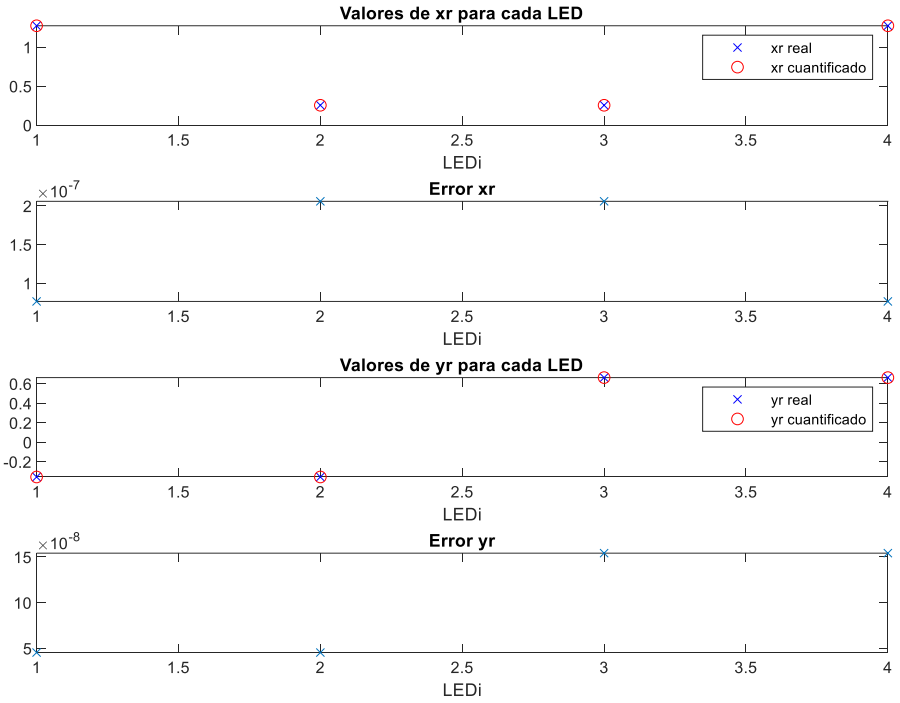

Figura 40. Representación de las coordenadas x_r e y_r en formato de coma fija Q2.22, junto con los errores cometidos.

- **Con 24 bits, formato Q6.18**

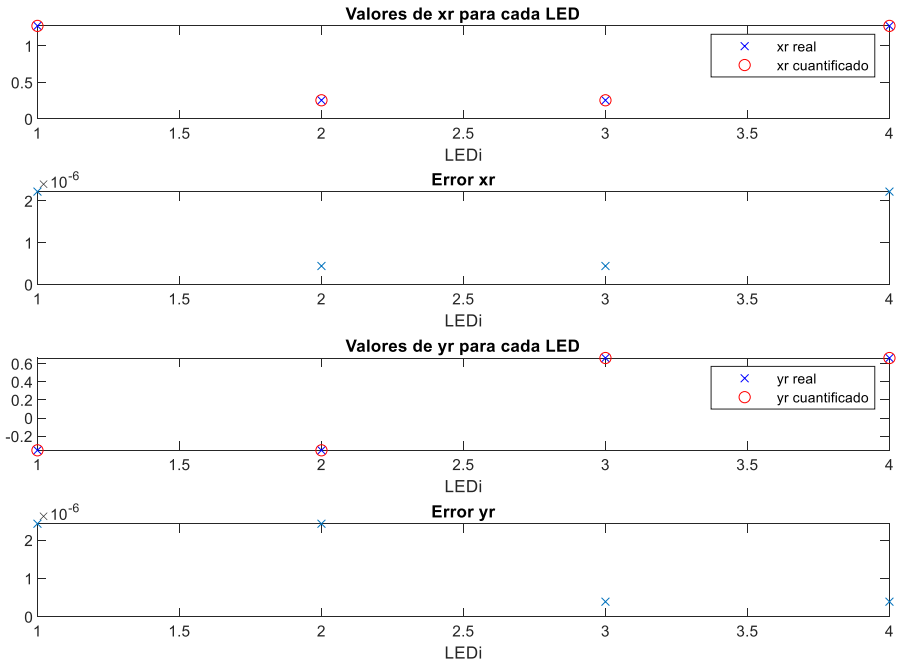

Figura 41. Representación de las coordenadas x_r e y_r en formato de coma fija Q6.18, junto con los errores cometidos.

- **Con 22 bits, formato Q6.16**

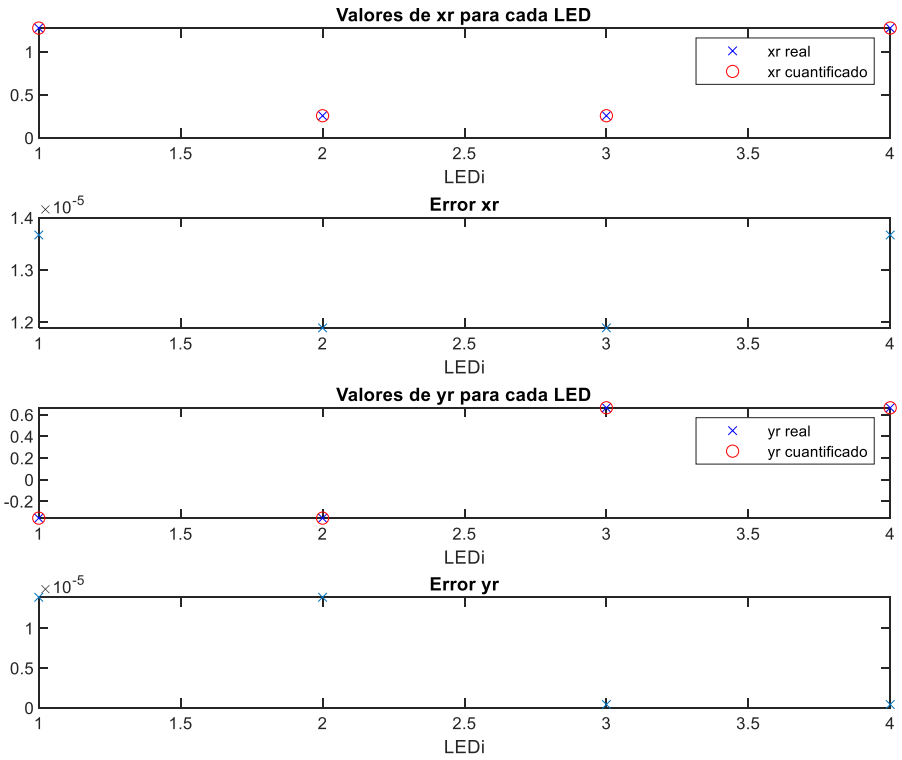

Figura 42. Representación de las coordenadas x_r e y_r en formato de coma fija Q6.16, junto con los errores cometidos.

- **Con 22 bits, formato Q2.20**

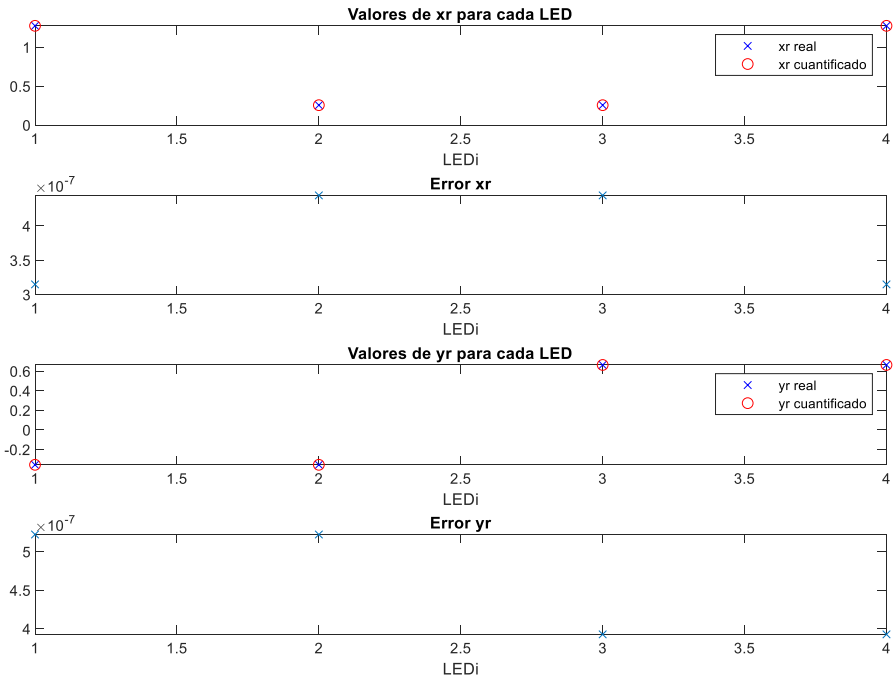

Figura 43. Representación de las coordenadas x_r e y_r en formato de coma fija Q2.20, junto con los errores cometidos.

- **Con 16 bits, formato Q6.10**

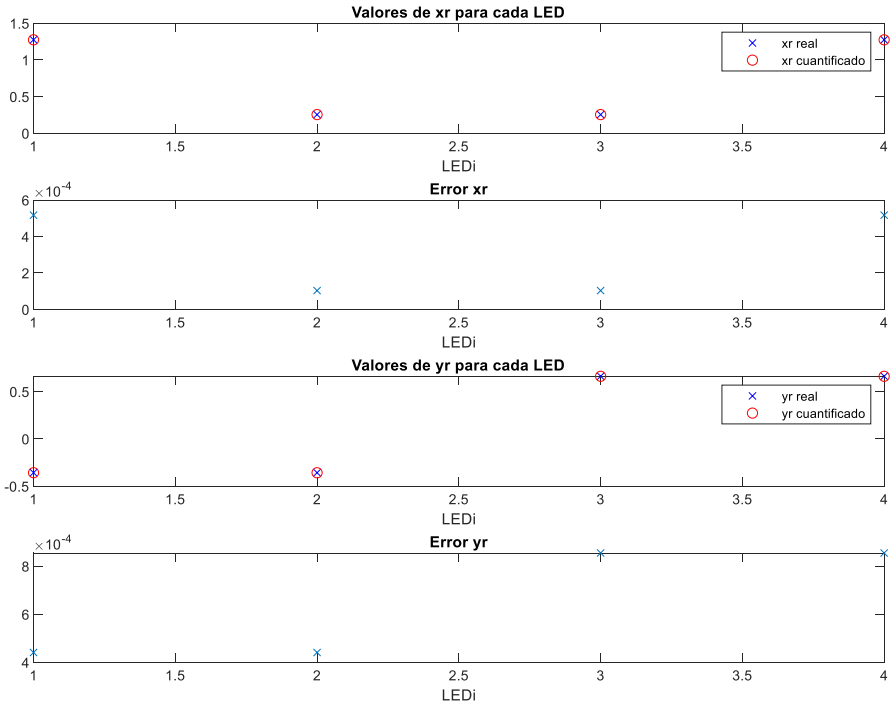

Figura 44. Representación de las coordenadas x_r e y_r en formato de coma fija Q6.10, junto con los errores cometidos.

- **Con 16 bits, formato Q4.12**

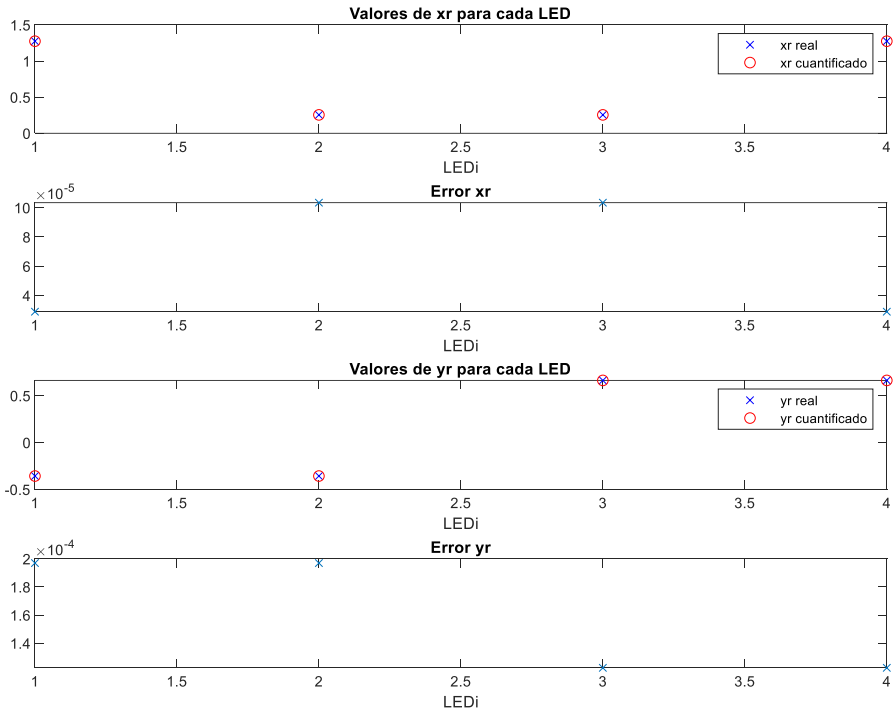

Figura 45. Representación de las coordenadas x_r e y_r en formato de coma fija Q4.12, junto con los errores cometidos.

- **Con 16 bits, formato Q2.14**

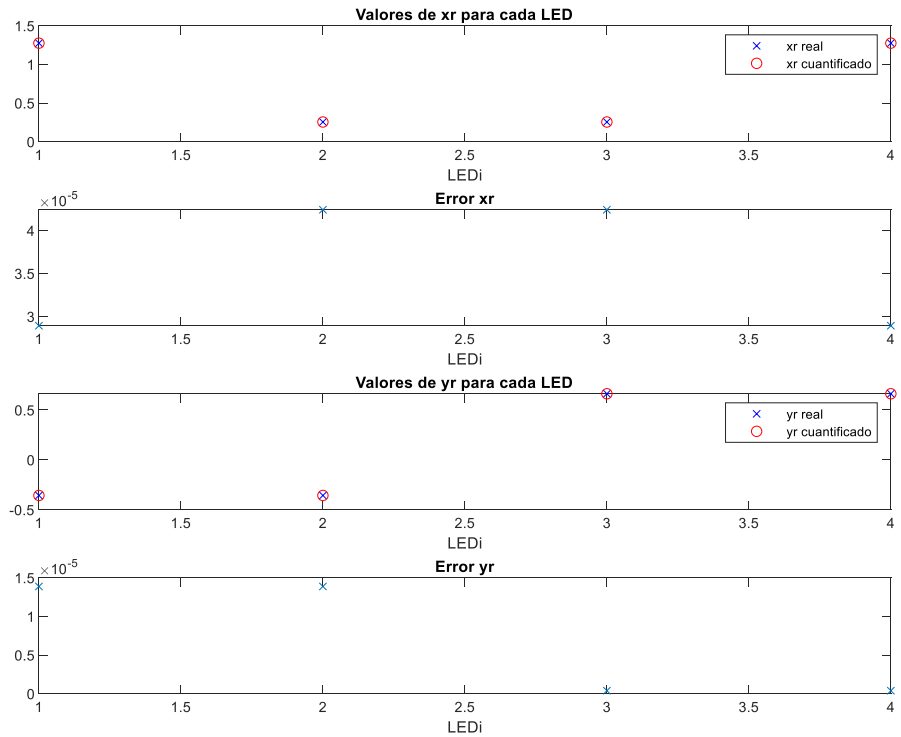

Figura 46. Representación de las coordenadas x_r e y_r en formato de coma fija Q2.14, junto con los errores cometidos.

Siguiendo el mismo razonamiento que en el punto anterior y observando la Tabla 10, resumen de los errores máximos relativos de las representaciones de las coordenadas con diferentes formatos de coma fija, se concluye que las coordenadas del punto de impacto tendrán una longitud de 16 bits de los cuales 14 representarán la parte decimal y 2 la parte entera.

Formato de representación	Error de la coordenada x_r	Error de la coordenada y_r
Q2.22	1.6159e-5%	2.3255e-5%
Q6.18	1.743e-5%	3.6647e-4%
Q6.16	0.0011%	0.0021%
Q2.20	3.4859e-5%	7.8768e-5%
Q6.10	0.0406%	0.1289%
Q4.12	0.0081%	0.0297%
Q2.14	**0.0033%**	**0.0021%**

Tabla 10. Errores máximos relativos de las coordenadas x_r e y_r en función del formato de representación en coma fija.

Capítulo 4

4. Descripción de la arquitectura propuesta

Tras el análisis del proceso de cálculo de las coordenadas de los puntos de impacto de cada LED en la superficie del fotorreceptor QADA y la adecuación de las variables y procesos implicados, se tiene una idea más clara de la estructura que puede adoptar la arquitectura a diseñar. Para el desarrollo de la misma se ha utilizado el entorno Xilinx VIVADO, donde se han diseñado y simulado los diferentes módulos, así como la arquitectura completa para verificar el correcto funcionamiento. Se ha utilizado el lenguaje de descripción hardware VHDL.

Atendiendo al carácter repetitivo de las operaciones realizadas sobre las diferentes señales de salida del fotorreceptor y los sucesivos resultados de las mismas, se ha propuesto la paralelización de dichas operaciones para cada una de las señales, aprovechando así la capacidad de concurrencia que tienen los sistemas hardware. Con esta premisa se ha esbozado el diagrama de bloques mostrado en la Figura 47. Se observa en él una entidad global llamada *general*, cuya arquitectura está constituida por diferentes módulos. En primer lugar, se tienen tres modelos VHDL con idéntica funcionalidad: *canal codevsum*, *canal codevlr* y *canal codevbt,* pero cada uno de ellos trata una de las tres señales de salida del QADA. Dichos módulos realizan las operaciones de correlación y la posterior detección de los picos máximos y mínimos. A partir de ellos y con la ayuda de diferentes componentes que se especificarán en los siguientes puntos, el módulo *general* obtiene las coordenadas de los puntos de impacto en el fotorreceptor QADA para cada LED.

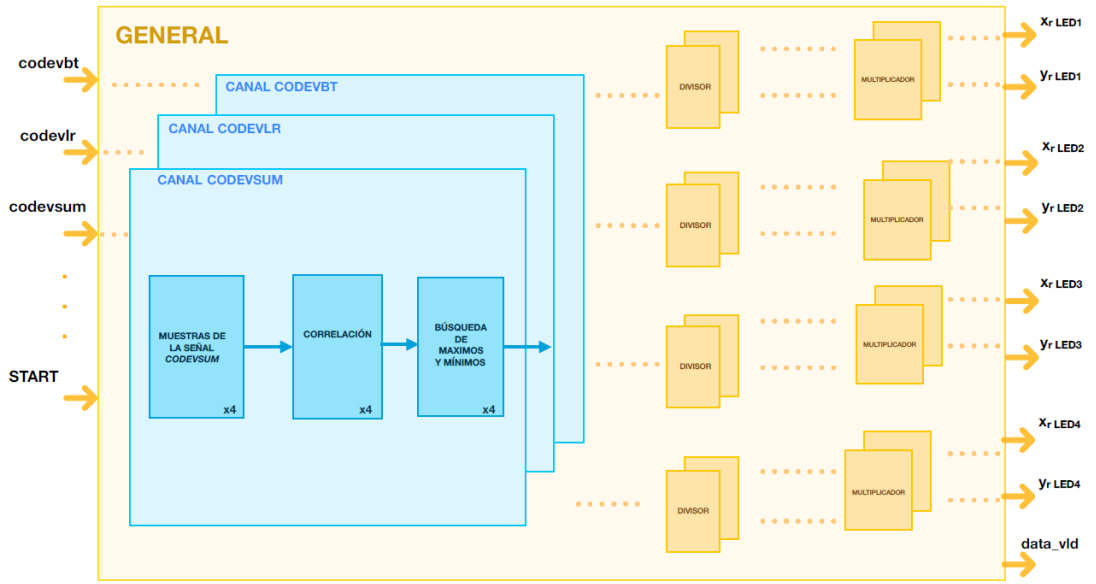

Figura 47. Diagrama de bloques simplificado de la arquitectura propuesta.

En los siguientes puntos se van a ir explicando por separado los diferentes módulos que conforman la arquitectura completa, aportando simulaciones funcionales y diagramas de bloques que complementen y faciliten la comprensión. Para la realización de las simulaciones funcionales, en este punto, no se han utilizado las muestras de las señales experimentales aportadas por [1], sino que se han empleado otros valores a modo de ejemplo que facilitan la comprobación del correcto funcionamiento.

4.1. Descripción de la entidad *CANAL*

La arquitectura propuesta contiene tres módulos *canal*, que son: *canal codevsum*, *canal codevlr* y *canal codevbt*. Cada módulo está constituido por cuatro componentes idénticos llamados *correlador*, otros cuatro componentes idénticos llamados *comparador* y una memoria RAM, como puede observarse en la Figura 48. Para cada uno de los módulos *canal*, los componentes *correlador* se encargan de correlar una de las señales de salida del QADA, con uno de los cuatro códigos emitidos por los LEDs. Por otro lado, cada componente *comparador* realiza la búsqueda de picos máximos del resultado de correlación aportado por uno de los componentes *correlador*. Estando constituidos los tres módulos *canal* por los mismos componentes se consigue una idéntica funcionalidad, diferenciándose unos de otros en las señales de entrada, como se muestra en la siguiente Figura 49.

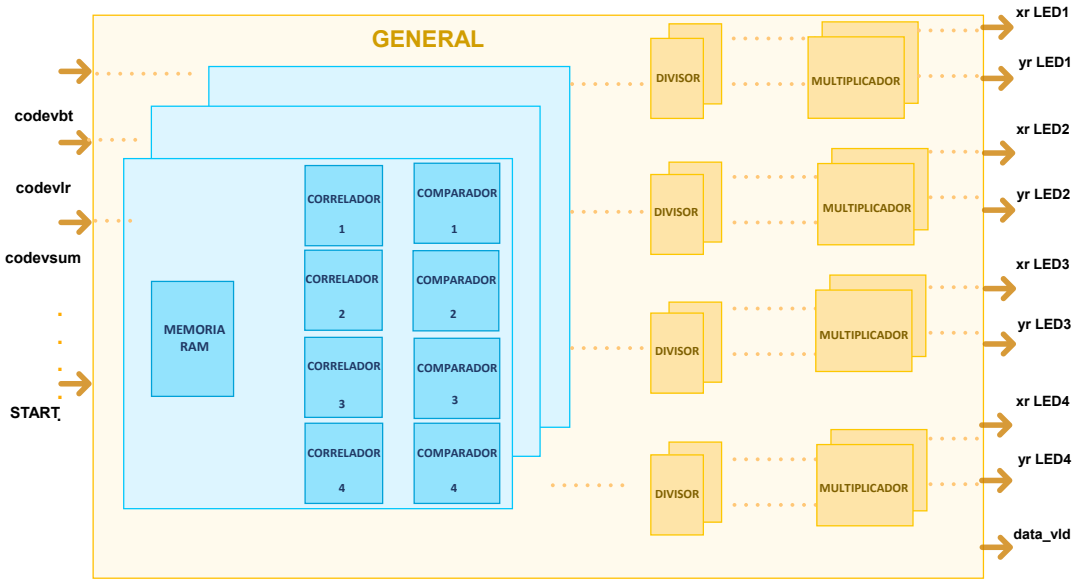

Figura 48. Estructura simplificada de los módulos canal propuestos.

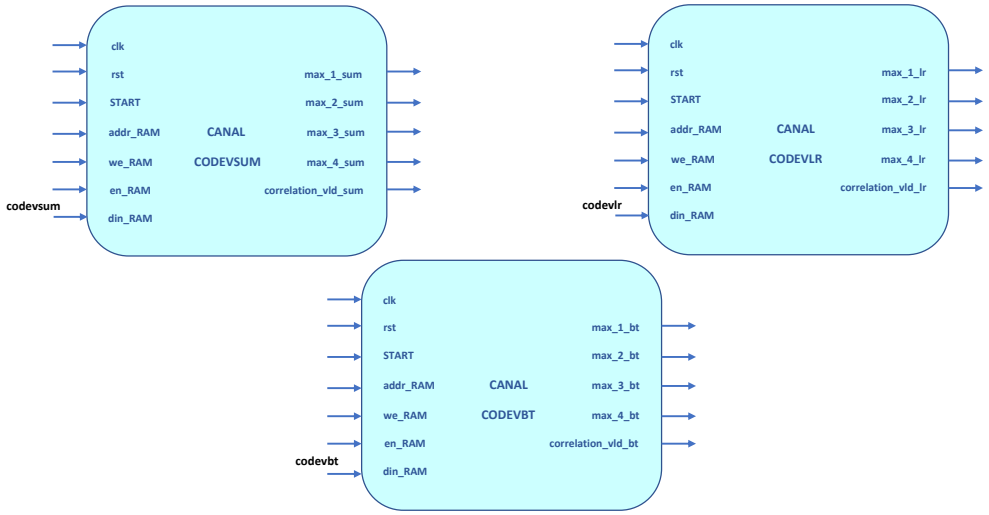

Figura 49. Puertos de los módulos canal codevsum, codevbt y codevlr.

En los siguientes puntos se explican los diferentes componentes que conforman los módulos *canal*. Como se ha mencionado anteriormente, dichos módulos tienen la misma estructura interna, por lo cual se utilizarán simulaciones y ejemplos correspondientes a uno de los tres canales, pudiéndose extrapolar los resultados al resto.

4.1.1. Memoria RAM

Un módulo *canal* necesita una memoria para almacenar las muestras de una de las señales del fotorreceptor QADA entregadas por el ADC. Dado que se va a realizar la correlación de dichas señales con los diferentes códigos emitidos por los LEDs, es necesario que todas las muestras

estén disponibles durante todo el periodo que dure el proceso de correlación. Además, el sistema de posicionamiento ha de poder muestrear las señales del fotorreceptor QADA a demanda, para poder así actualizar la posición del receptor. Esto implica que la memoria debe tener la capacidad de sobrescribir información y poder ser leída cada cierto periodo de tiempo, hasta ahora indefinido. Por todo lo comentado anteriormente, se propone el uso de una memoria Simple Dual Port RAM, pues solo se escribirán datos por parte del ADC y solo se leerán éstos por parte de los módulos *correlador*. Para el modelado de dicha memoria en VHDL se ha hecho uso de los cores que ofrece la plataforma Xilinx VIVADO. En la Figura 50, se muestran los puertos de la memora RAM utilizada, donde puede observarse que solo se puede escribir a través del puerto A y solo se puede leer mediante el puerto B.

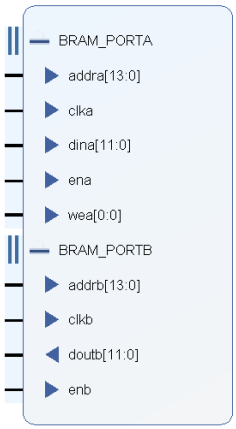

Figura 50. Puertos del core BRAM utilizado.

Por otro lado, como se propuso en el punto anterior, cada muestra de la señal se va a representar en coma fija con un total de 12 bits. Asimismo, por el caso experimental estudiado en [1] y utilizado para el desarrollo de la arquitectura de este proyecto, se sabe que el total de muestras de las señales tomadas por el ADC es de 11540. Por consiguiente, se ha configurado dicha RAM con un ancho de 12 bits y una profundidad de 16K ($\log_2(11540) \approx 14 \Rightarrow 2^{14} = 16K$).

El motivo por el cual se cuenta con dicho número de muestras, 11540, está relacionado con las 11510 muestras que constituyen los códigos emitidos por los LEDs. Por lo cual, si lo que se quiere es detectar dichos códigos en el receptor habrá que muestrear, mínimo, tantas veces como muestras tenga cada código.

4.1.2. Descripción del componente correlador

Para el desarrollo de la arquitectura VHDL que realiza la correlación de una de las señales de salida del QADA con uno de los cuatro códigos emitidos por los LEDs, se ha partido de la operación básica del proceso que es la multiplicación de dos valores y la posterior acumulación de los sucesivos resultados de la misma. Los coeficientes de la multiplicación son, por un lado, una muestra de una de las señales del QADA y, por otro, una muestra de un código LED. Como se comentó anteriormente las señales del QADA se encuentran almacenadas en una memoria RAM, mientras que los códigos emitidos por cada LED están en una memoria ROM. En este caso, se ha elegido este tipo de memoria porque las muestras de los códigos no van a ser modificadas en ningún punto del proceso y solo será necesario leerlas. Sabiendo, por el caso experimental

estudiado en [1] y utilizado para el desarrollo de la arquitectura de este proyecto, que son 11510 muestras de 12 bits cada una, las que conforman cada código de un LED, dicha memoria ROM ha sido configurada con un ancho de 12 bits y 16K de profundidad. Además, igual que la memoria RAM, la memoria ROM se ha implementado usando un core de VIVADO. En la Figura 51 se muestran los puertos de la memoria ROM.

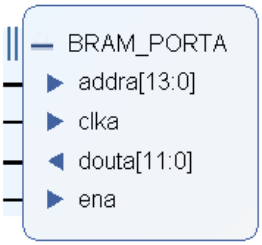

Figura 51. Puertos del core de la memoria ROM utilizado.

El circuito lógico modelado en VHDL, que realiza la operación básica de multiplicación y acumulación, se muestra en la Figura 52. Puede observarse que se compone de las dos memorias mencionadas, un multiplicador, un registro y un sumador que opera sobre el resultado de la multiplicación y la salida del registro.

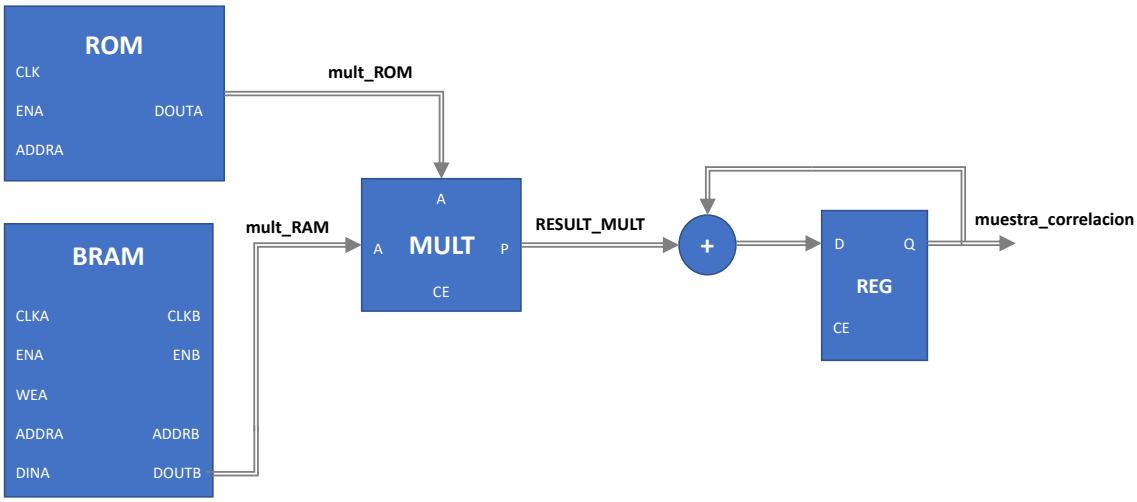

Figura 52. Diagrama de bloques de la parte del módulo correlador dedicada a la multiplicación y acumulación de muestras.

Para el modelado en VHDL del circuito, la función de multiplicación es realizada por uno de los cores que ofrece Xilinx VIVADO y ha sido configurado para que multiplique valores de 12 bits con signo, mientras que el acumulador ha sido diseñado con VHDL mediante un registro. En la siguiente figura se muestran los puertos del multiplicador utilizado.

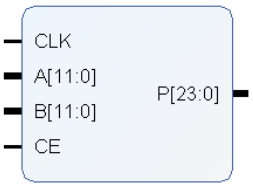

Figura 53. Puertos del core multiplicador utilizado.

Como apoyo a la explicación de la arquitectura del módulo de correlación propuesto, se ha desglosado de forma visual y simplificada el procedimiento seguido para el cálculo de una correlación en la Figura 54. Se ha representado cada memoria como un array, donde la posición S_0, del array que representa la memoria RAM, contiene la primera muestra de una de las señales de salida del QADA y la posición C_0, del array que representa a la memoria ROM, contiene la última muestra de uno de los códigos emitidos por los LEDs. De forma visual, puede explicarse el proceso como el desplazamiento, de izquierda a derecha, del array correspondiente a la ROM sobre el array de RAM. En cada desplazamiento se multiplican los elementos de las posiciones coincidentes y después se suman los resultados. Así, en el ejemplo, la segunda muestra de correlación se calcula multiplicando el valor correspondiente a la primera posición de la RAM, S_0, con el valor correspondiente a la segunda posición de la ROM, C_1, sumado a la multiplicación del elemento de la segunda posición de la RAM, S_1, con el valor de la primera posición de la ROM, C_0. Nótese que se trata de un ejemplo simplificado; en el caso real la memoria RAM contiene 11540 muestras y no 5, al igual que la ROM tiene 11510 muestras y no 3. Lo que sí se ha mantenido es la relación entre el número de muestras, teniendo mayor número de elementos la memoria RAM que la memoria ROM.

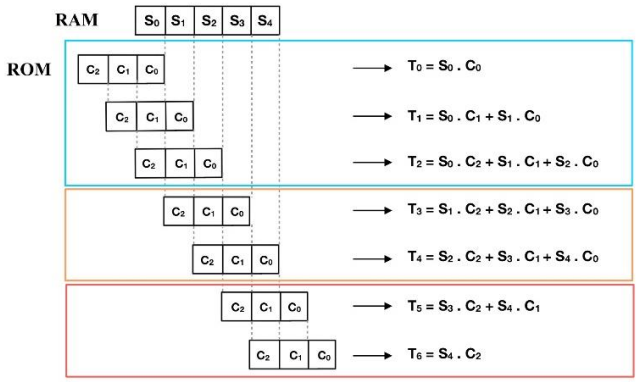

Figura 54. Representación gráfica del proceso de correlación.

Atendiendo de nuevo a la Figura 54, puede observarse que el módulo diseñado hasta ahora tiene la capacidad de realizar las operaciones necesarias para el cálculo de una muestra de correlación, pero es preciso desarrollar un sistema que gestione qué muestras han de multiplicarse, cuándo y en qué orden. Por ello se ha desarrollado un modelo que genera las direcciones de memoria correspondientes a las muestras necesarias en cada momento y una máquina de estados que gestiona dicha generación de direcciones.

La máquina de estados diseñada consta de un total de cuatro estados, pudiéndose diferenciar tres de ellos en la Figura 54. Visualmente, el segundo estado se corresponde con la parte recuadrada en azul, es decir, desde que la primera posición del array de RAM S_0 coincide con la

primera posición del array de ROM C_0, hasta que la primera posición de la RAM coincide con la última posición del array de ROM C_2. El tercer estado, representado por la parte marcada en naranja, se da cuando todas las posiciones del array de ROM coinciden con posiciones del array de RAM hasta que la última posición de la RAM S_4 coincide con la primera posición del array de ROM C_0. Por último, el cuarto estado representado en rojo se mantiene desde que la primera posición del array de ROM C_0 deja de coincidir con la última posición del array de RAM hasta que no existe coincidencia alguna entre posiciones de ambos arrays. A continuación, se detalla cómo se ha llevado esta percepción visual al diseño lógico en VHDL.

Los estados lógicos definidos para la gestión de la generación de direcciones de las memorias RAM y ROM son:

- Primer estado: ***reposo***. Es el estado en el que se encuentra el sistema tras un reinicio o encendido. También puede llegarse a él cuando el *correlador* termina de calcular la última muestra de correlación. Para cambiar al siguiente estado, *first*, la condición de transición que ha de cumplirse es que una señal externa llamada *START* se active. Dicha señal marca el inicio del proceso de cálculo de las coordenadas de los puntos de incidencia para cada LED.
- Segundo estado: ***first***. El sistema llega al estado *first* cuando la señal START se activa y sale de él cuando el número de muestras de correlación calculadas coincide con el número de muestras almacenadas en la memoria ROM.
- Tercer estado: ***second***. A este estado se llega cuando se cumple la condición de transición en el estado *first* y se sale de él, al estado *third*, cuando el número de muestras de correlación calculadas coincide con el número de muestras almacenadas en la memoria RAM.
- Cuarto estado: ***third***. Se sale de este estado cuando se han calculado todas las muestras de correlación, es decir, tantas muestras como resultado da la suma del número de muestras de la memoria RAM, más el número de muestras de la memoria ROM menos 1. De este estado se pasa al primer estado, *reposo,* quedando el sistema a la espera de tener que iniciar de nuevo el proceso de cálculo de correlaciones.

En la Figura 55 se muestran gráficamente los diferentes estados explicados y las posibles transiciones entre ellos.

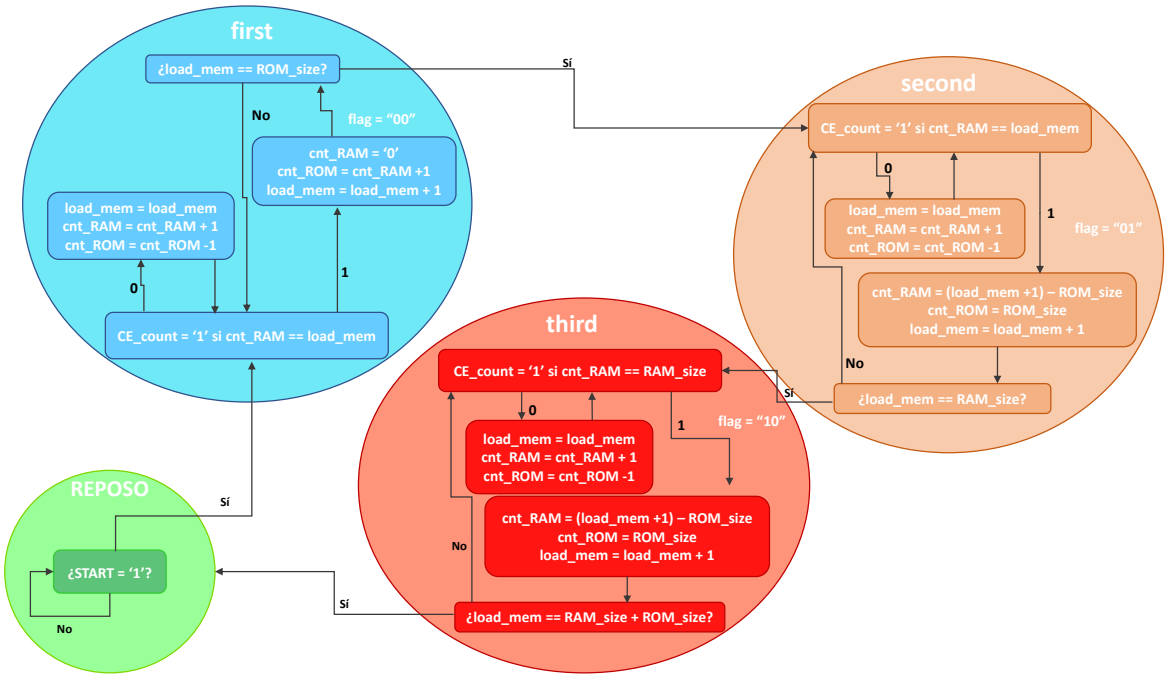

Figura 55. Diagrama de estados que gestiona la generación de direcciones para las memorias RAM y ROM.

Hasta ahora, se han mencionado parámetros como el número de muestras de correlación calculadas, una señal externa *START, flag,* … etc., las cuáles son generadas y utilizadas en el bloque que se encarga de generar las direcciones de las memorias. Dicho sistema consta principalmente de dos módulos, concretamente dos contadores, uno de ellos encargado de contar el número de muestras de correlación calculadas, y el otro encargado de generar indirectamente las direcciones de la memoria RAM y ROM.

- El contador de muestras de correlación calculadas lleva la cuenta sobre la señal *load_mem* incrementándola siempre que la señal *CE_count1* valga uno. Esta última señal se activa siempre que, por medio de una sentencia condicional, se detecte que el cálculo de una muestra de correlación ha sido finalizado. Para ello se compara la señal *cnt_RAM* con la señal CE_count1 y *RAM_size*, dependiendo del estado en el que se encuentre el sistema:
    - Para los estados *first* y *second*, el sistema considera que el cálculo de la muestra de correlación ha finalizado cuando la dirección de la memoria RAM sea igual al número de muestras de correlación calculadas, es decir, igual a *load_mem*. Analizando de nuevo la Figura 54 se observa que, para cada muestra de correlación, el último valor de la posición de memoria RAM a la que se accede coincide con el número de muestra de correlación. Esto se refleja en la siguiente figura, que resalta en gris la coincidencia entre el subíndice de la muestra de correlación T_x y el de la posición de la RAM C_y:

$$T_0 = S_0 \cdot C_0$$
$$T_1 = S_0 \cdot C_1 + S_1 \cdot C_0$$
$$\dots$$
$$T_4 = S_2 \cdot C_2 + S_3 \cdot C_1 + S_4 \cdot C_0$$

Figura 56. Análisis del cálculo de las muestras de correlación, donde se observa que el número de muestra coincide con la posición de la memoria RAM.

Por lo tanto, siguiendo la analogía del ejemplo de la Figura 54, puede decirse que la señal *load_mem* lleva la cuenta de la variable *x.*

- Cuando el sistema se encuentra en el estado *third*, el cálculo de las muestras de correlación termina cuando la dirección de memoria RAM es igual a la posición que contiene la última muestra de una de las señales del fotorreceptor QADA, es decir, igual a la constante *RAM_size*. En el modelo VHDL desarrollado para el *correlador* se han definido dos constantes, *RAM_size* y *ROM_size*, las cuales contienen el número total de muestras de las señales de salida del QADA menos uno y el número total de muestras de los códigos emitidos por los LED menos uno, respectivamente.
  Para el ejemplo de la Figura 54, *RAM_size* sería igual a 4 y *ROM_size* igual a 2. Se observa entonces en la Figura 57 que, cuando el subíndice de la posición de memoria de la RAM coincide con *RAM_size*, ya se ha realizado la última multiplicación necesaria para el cálculo adecuado de la muestra de correlación.

$$T_5 = S_3 \cdot C_2 + S_4 \cdot C_1$$
$$T_6 = S_4 \cdot C_2$$

Figura 57. Análisis del cálculo de las muestras de correlación, donde se observa que la última posición de memoria RAM a la que se accede coincide con RAM_size.

- El contador que genera, indirectamente, las direcciones de las memorias, incrementa y decrementa las señales *cnt_RAM* y *cnt_ROM*, respectivamente, restableciendo sus valores en función de las señales *CE_count1* y *flag*.
  - Cuando el sistema se encuentra en el estado *first*, la variable *flag* vale "00". En este caso, cuando se detecte que se ha finalizado el cálculo de una muestra de correlación, *CE_count1* = '1', la variable *cnt_RAM* se pondrá a 0 mientras que *cnt_ROM* será igual al último valor de *cnt_RAM* más uno.
    De nuevo, atendiendo a la Figura 54, puede evidenciarse este razonamiento. Para el caso de la tercera muestra T_2, a la primera posición de memoria RAM a la que se accede es la cero, mientras que la primera posición de memoria ROM leída es la dos, que coincide con la última posición de la RAM leída S_1, en el cálculo de la muestra anterior más uno C_2.
  - Cuando el sistema está en el estado *second* o *third*, la variable *flag* vale "01" y "10" respectivamente. Cuando *CE_count1 = '1'*, *cnt_RAM* toma el valor de *load_mem* + 1 – *ROM_size*. Por último, *cnt_ROM* se actualiza con el valor de *ROM_size*.

A partir de las señales *cnt_RAM* y *cnt_ROM* se obtienen las direcciones de las memorias RAM y ROM como se muestra en la Figura 58:

```
direccion_ ROM <= ROM_size - cnt_ROM;
direccion_RAM <= cnt_RAM;
```

Figura 58. Fragmento de código VHDL del módulo correlador que muestra la generación de las direcciones de memoria RAM y ROM.

En la Figura 59, se muestran los puertos de un módulo *correlador* y la estructura interna de su arquitectura.

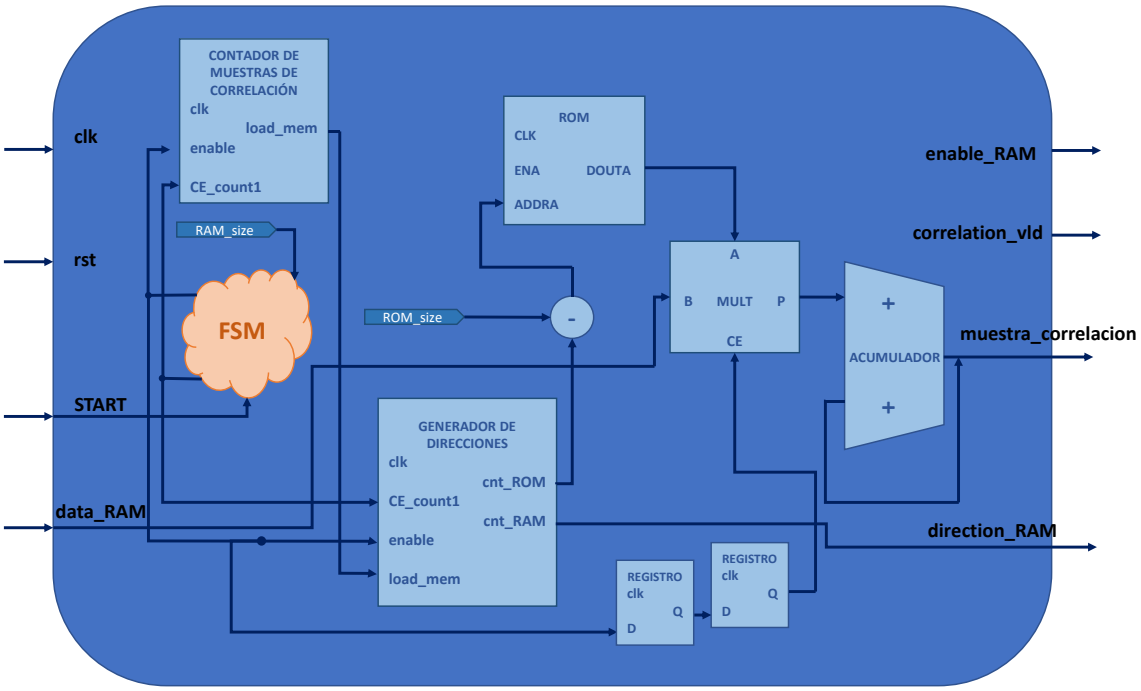

Figura 59. Diagrama de bloques completo de un módulo correlador.

A continuación, se muestra una simulación funcional del módulo *correlador*. Como se comentó anteriormente, se han empleado valores que simplifican la tarea de verificación de la funcionalidad. Para ésta y las restantes simulaciones funcionales mostradas a lo largo de este punto 4, la memoria ROM contiene los valores [1,2,3] y la RAM [1,2,3,4,5]. La memoria ROM ha sido cargada por medio de un fichero .coe, mientras que los elementos de la RAM han sido escritos mediante un testbench. En él se ha instanciado el componente a verificar, el *correlador,* se ha habilitado la escritura en la memoria RAM y se han inyectado los datos de test almacenados en un fichero. Después, se ha activado la señal START dando lugar al inicio del cálculo de las correlaciones.

En la Figura 60, se muestran solo las señales utilizadas y generadas por el módulo *correlador*. Puede observarse como, tras la activación de la señal START, en *t* = 375 ns, se habilita la lectura de la memoria RAM con *enb* = '1'. Las direcciones de las memorias empiezan a generarse y se obtienen los valores correspondientes de la ROM y la RAM en las señales *douta* y *doutb,* respectivamente. En *t* = 395 ns, la señal *enable_FF_FF* habilita el multiplicador, que empieza a operar las muestras presentes en *douta* y *doutb,* mostrando los resultados en la señal *RESULT_MULT.* Posteriormente, en *t* = 405 ns, la señal *enable_acum* habilita el acumulador, que va sumando los resultados de las multiplicaciones y reflejando el resultado en la variable *result_sum_v*. Por último, en la señal *m_correlacion_1* pueden observarse los resultados de cada muestra de correlación: 3, 8, 14, 20, 26, 14 y 5. Una vez calculadas todas las muestras la señal *correlation_vld_sum* genera un pulso que indica que el proceso de correlación ha terminado. Puede comprobarse observando las señales *flag* y *state* cómo el sistema cambia de estado y con ello la señal *flag* se va actualizando.

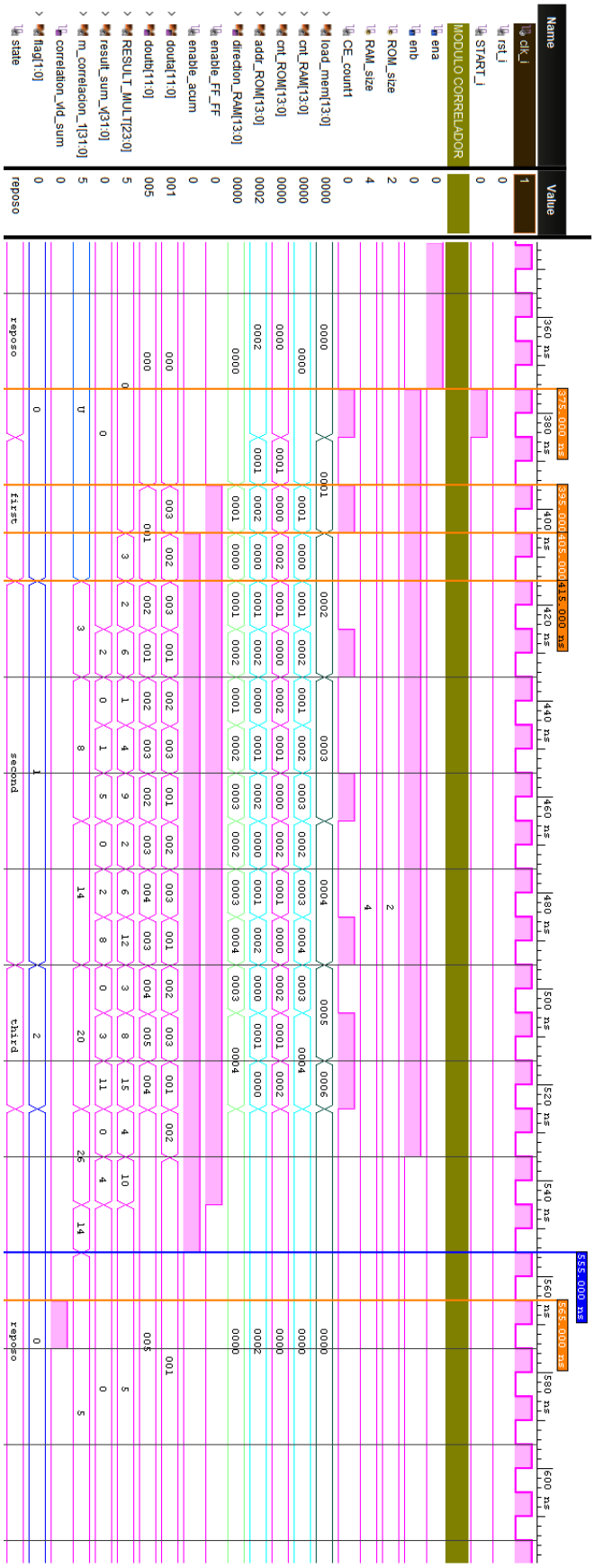

Figura 60. Simulación funcional del módulo correlador.

4.1.3. Propuesta de arquitectura de búsqueda de picos de correlación.

Como se observa en la Figura 59, un módulo *correlador* genera, entre otras, una señal de salida *muestra_correlacion,* que contiene el resultado de cada muestra de correlación realizada y una señal de validación, *correlation_vld*, que se activa a nivel alto cuando se tiene un valor correcto de correlación. A partir de estas señales, el módulo comparador va registrando y comparando las sucesivas muestras de correlación buscando valores máximos positivos o negativos. Este módulo sólo entrega a su salida el valor de una muestra de correlación cuando detecta que efectivamente se trata de un máximo. En la Figura 61 se ha representado la entidad del *comparador* junto con la arquitectura propuesta.

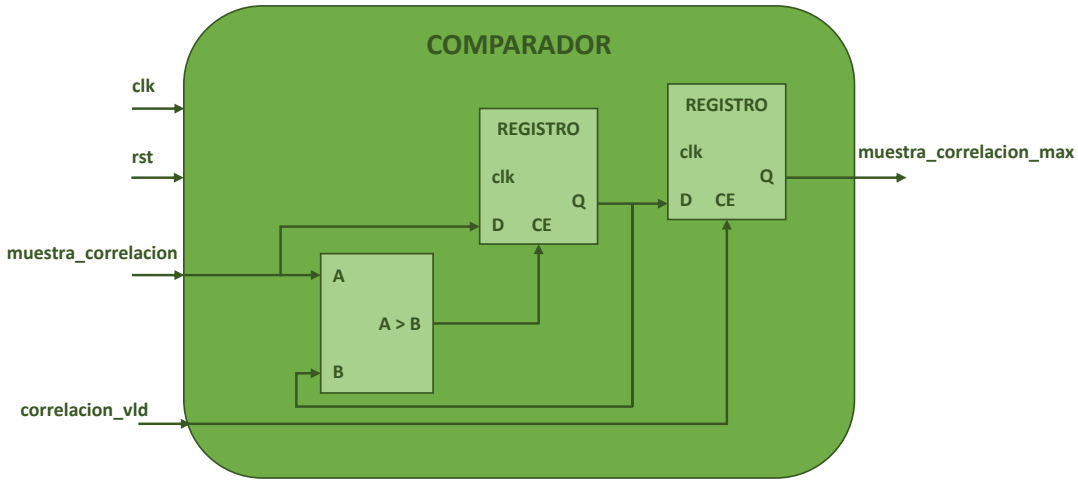

Figura 61. Diagrama de bloques del módulo comparador.

A continuación, en la Figura 62, se muestra la simulación funcional del módulo *correlador* explicada anteriormente y, además, se ha incluido el módulo *comparador*. Se observa la señal *out_absolute*, la cual refleja el valor absoluto de las muestras de correlación. La señal *CE_max* se activa a nivel alto cada vez que se detecta una muestra de correlación mayor que la anterior. Puede observarse como en t = 545 ns, tras registrarse la quinta muestra de correlación que contiene el valor máximo de correlación, *CE_max* pasa a valer '0'. Por último, cuando el módulo *correlador* notifica el fin del proceso de correlación con un pulso en la señal *correlation_vld_sum*, el comparador entrega el valor máximo de la correlación realizada, en t = 575 ns.

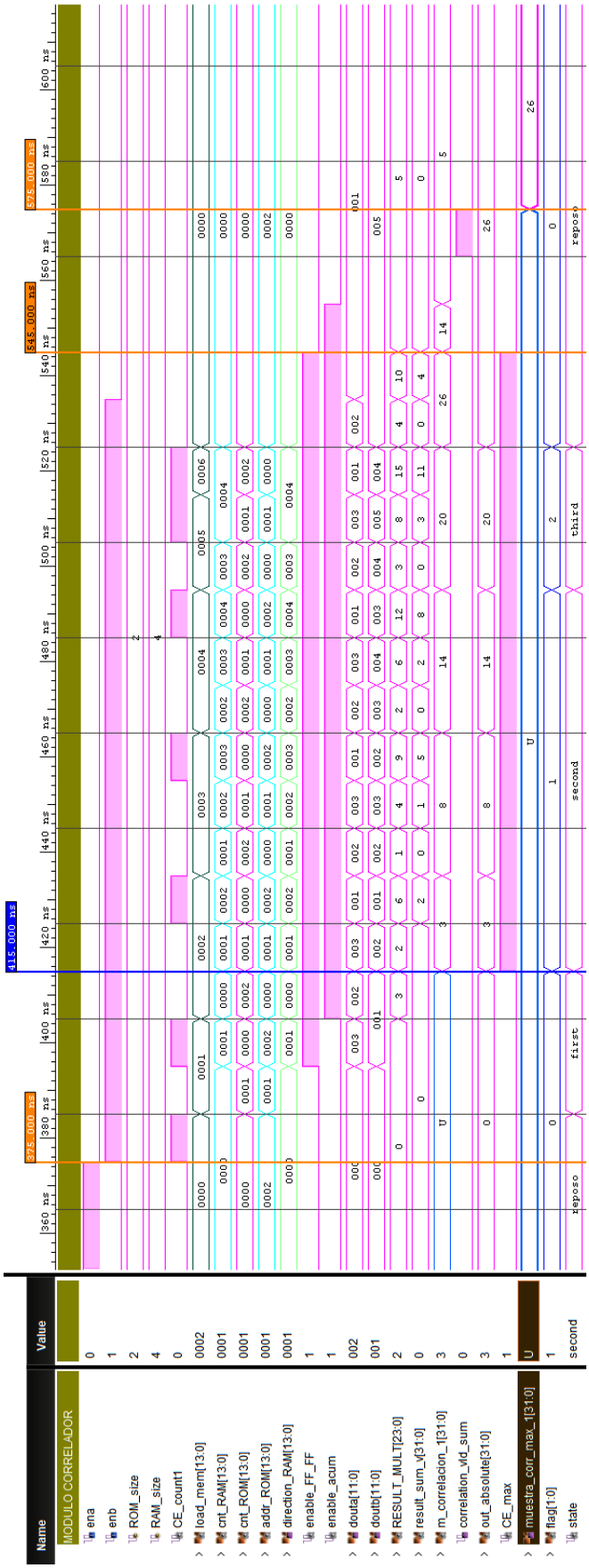

Figura 62. Simulación funcional del módulo correlador y comparador.

En resumen, el módulo *canal* propuesto está constituido por una memoria RAM, que contiene las muestras correspondientes a una de las tres señales de salida del QADA. Particularizando para el *canal codevsum*, las muestras de la señal *codevsum* están almacenadas en la memoria RAM. Dicha señal es correlada con cada uno de los códigos emitidos por los LEDs mediante los cuatro módulos *correlador*. Así, el primero de ellos correla la señal *codevsum* con el código emitido por el LED1, al mismo tiempo, el segundo *correlador* lo hace con el código emitido por el LED2, el tercero con el código del LED3 e igualmente el cuarto correlador con el código emitido por el LED4. Cada módulo *correlador* contiene una memoria ROM que almacena las muestras correspondientes a uno de los códigos emitidos por los LEDs. Por último, a cada *correlador* le sigue un *comparador* que realiza la búsqueda del valor máximo de cada correlación.

Para modelar los bloques *canal* en VHDL, primero se han modelado los módulos *correlador* y *comparador* por separado, se han simulado para comprobar el correcto funcionamiento, como se ha ido reflejando en cada simulación funcional aportada, y por último se han instanciado ambos módulos cuatro veces en cada uno de los tres módulos *canal*.

En la Figura 63 y Figura 64, se muestran las simulaciones funcionales de los canales *codevsum*, *codevbt* y *codevlr*, respectivamente. Se han representado sólo los puertos de entrada y salida de cada entidad. Se observa como los tres módulos llegan al mismo resultado, pues todas las memorias han sido cargadas con los mismos datos. Por último, en la Figura 65, se ha añadido un nuevo estímulo a la señal START, en t = 595 ns. Se observa que los valores máximos de las correlaciones no varían, lo que quiere decir que el proceso se realiza de nuevo sin errores.

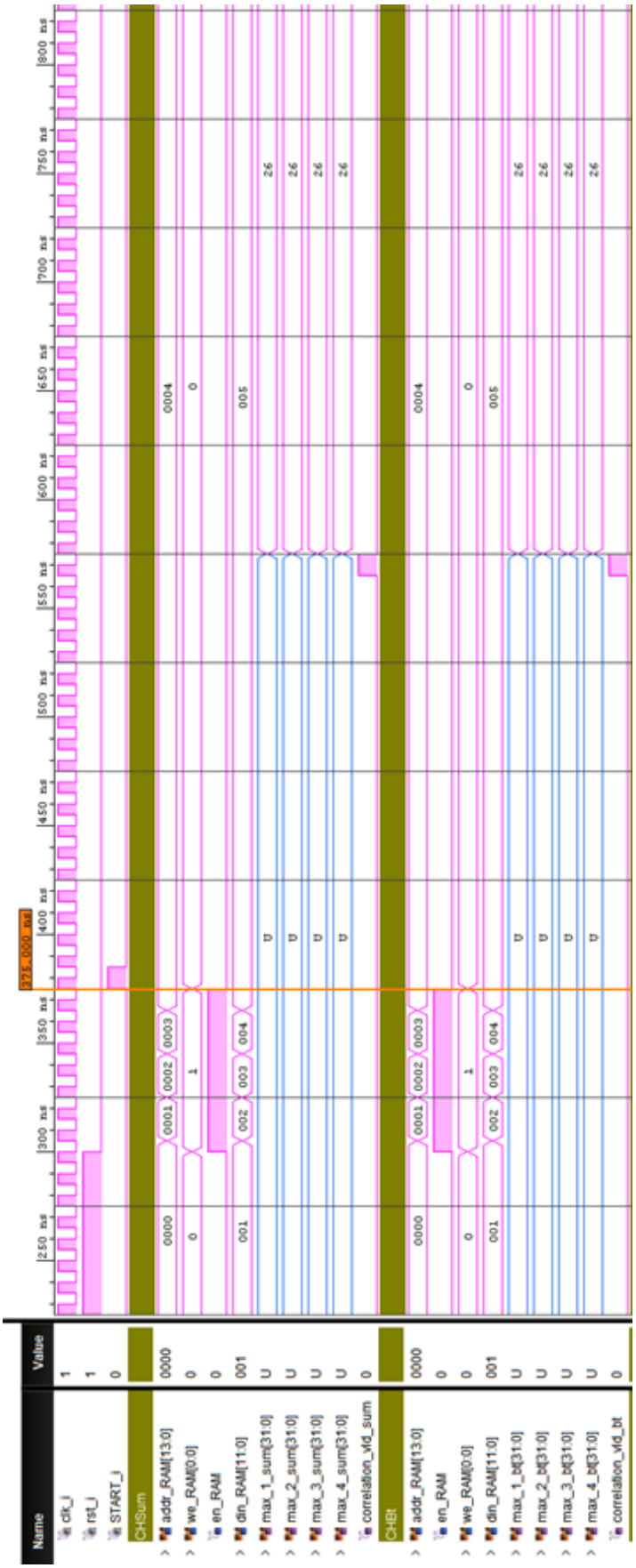

Figura 63. Simulación funcional de los módulos canal codevsum y canal codevbt.

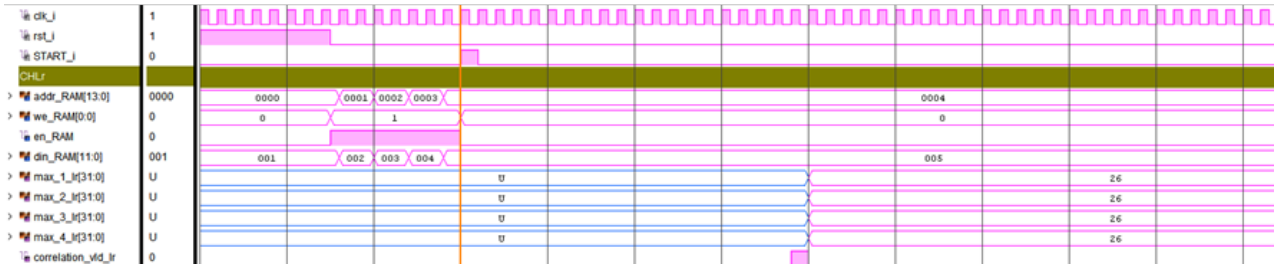

Figura 64. Simulación funcional del módulo canal codevlr.

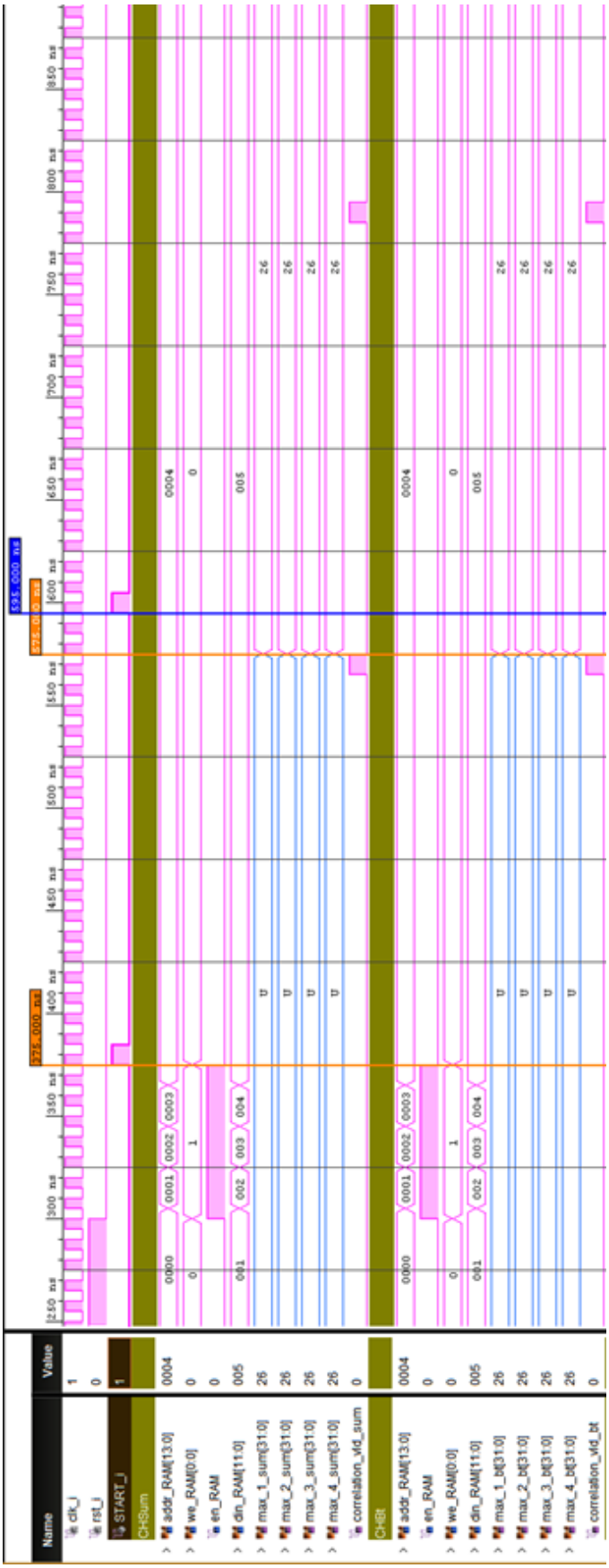

Figura 65. Simulación funcional de los módulos canal codevsum y canal codevbt, con dos pulsos de la señal START.

4.2. Descripción de la entidad GENERAL

El módulo *general* es el modelo VHDL que contiene la arquitectura capaz de calcular las coordenadas de los puntos de incidencia en el fotorreceptor para cada transmisor LED. Esto se ha conseguido instanciando, en dicha arquitectura, el módulo *canal* tres veces, uno para cada señal de salida del QADA, obteniéndose así los máximos y mínimos de las correlaciones de dichas señales con cada código LED emitido, como se explicó en el apartado anterior. En la Figura 66 se muestran las señales de entrada y salida que constituyen la entidad *general*. Los puertos *addr_RAM*, *we_RAM* y *en_RAM* son señales de control utilizadas para la escritura de las señales de salida del QADA en las tres memorias RAM de los bloques *canal*. Estas señales son comunes a los tres bloques porque, en esta propuesta, la escritura en memoria de las señales es simultánea. Por otro lado, los puertos *din_RAM_codevsum*, *din_RAM_codevlr* y *din_RAM_codevbt* proporcionan las muestras correspondientes a las señales *codevsum*, *codevlr* y *codevbt* respectivamente, a los diferentes módulos *canal* para su escritura en memoria. Por último, la señal START inicia el proceso de cálculo de las coordenadas de los puntos de impacto siempre que se active a nivel alto.

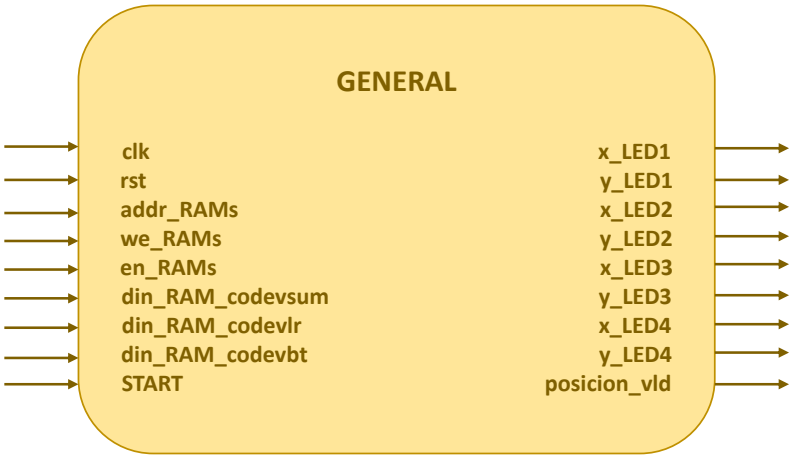

Figura 66. Puertos de entrada y salida del módulo general

4.2.1. Cálculo de las relaciones p_x y p_y

Una vez obtenidos los valores máximos de las correlaciones se pueden calcular las relaciones p_x y p_y para cada uno de los cuatro transmisores LED. Como se definió en las ecuaciones (1) y (2), para el cálculo de p_x han de dividirse los resultados máximos de la correlación de la señal *codevlr* con cada código LED entre los valores máximos de correlación de la señal *codevsum* con cada código LED y, para el cálculo de p_y, han de dividirse los resultados máximos de la correlación de la señal *codevbt* con cada código LED entre los valores máximos de correlación de la señal *codevsum* con cada código LED.

Para modelar este cálculo en VHDL se ha hecho uso de los cores que ofrece VIVADO para la división de los diferentes valores y se han configurado para que dividan números con signo de 32 bits. En la Figura 67 se muestra la entidad del divisor utilizado, donde pueden observarse sus puertos de entrada y salida. El módulo necesita unas señales de validación, *s_axis_divisor_tvalid* y *s_axis_dividend_tvalid*, que indican que los valores presentes en los pines *s_axis_divisor_tdata* y *s_axis_dividend_tdata* en un determinado momento son válidos y, por tanto, son con los que

ha de operar. Se han utilizado como señales de validación de los ocho cores divisores a instanciar, las tres señales de validación de correlación de los tres módulos *canal* registradas, es decir, *sum_vld_reg*, *bt_vld_reg* y *lr_vld_reg*. De no registrarse estas señales, los divisores operarían con dividendos y divisores erróneos o incluso con valores no definidos. La Figura 68 y Figura 69 contienen esquemas donde se muestra cómo se han conectado entre sí los diferentes módulos. Se ha separado la representación en dos figuras para una mejor visualización de las conexiones, solo existe un módulo *canal codevsum* dentro de la arquitectura general.

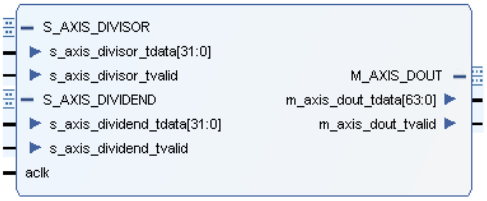

Figura 67. Puertos de entrada y salida del core divisor utilizado.

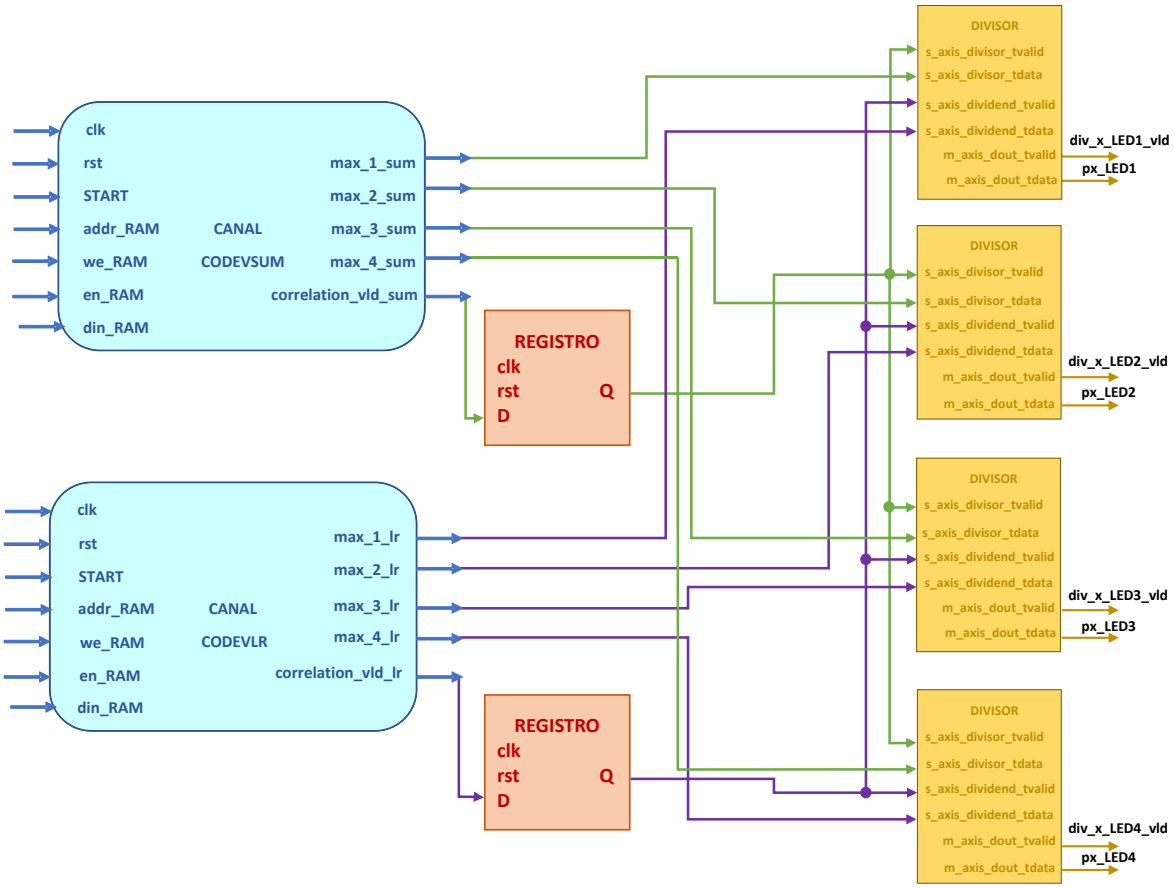

Figura 68. Diagrama de bloques que muestra la conexión entre los módulos canal y los divisores para el cálculo de las relaciones p_x y p_y.

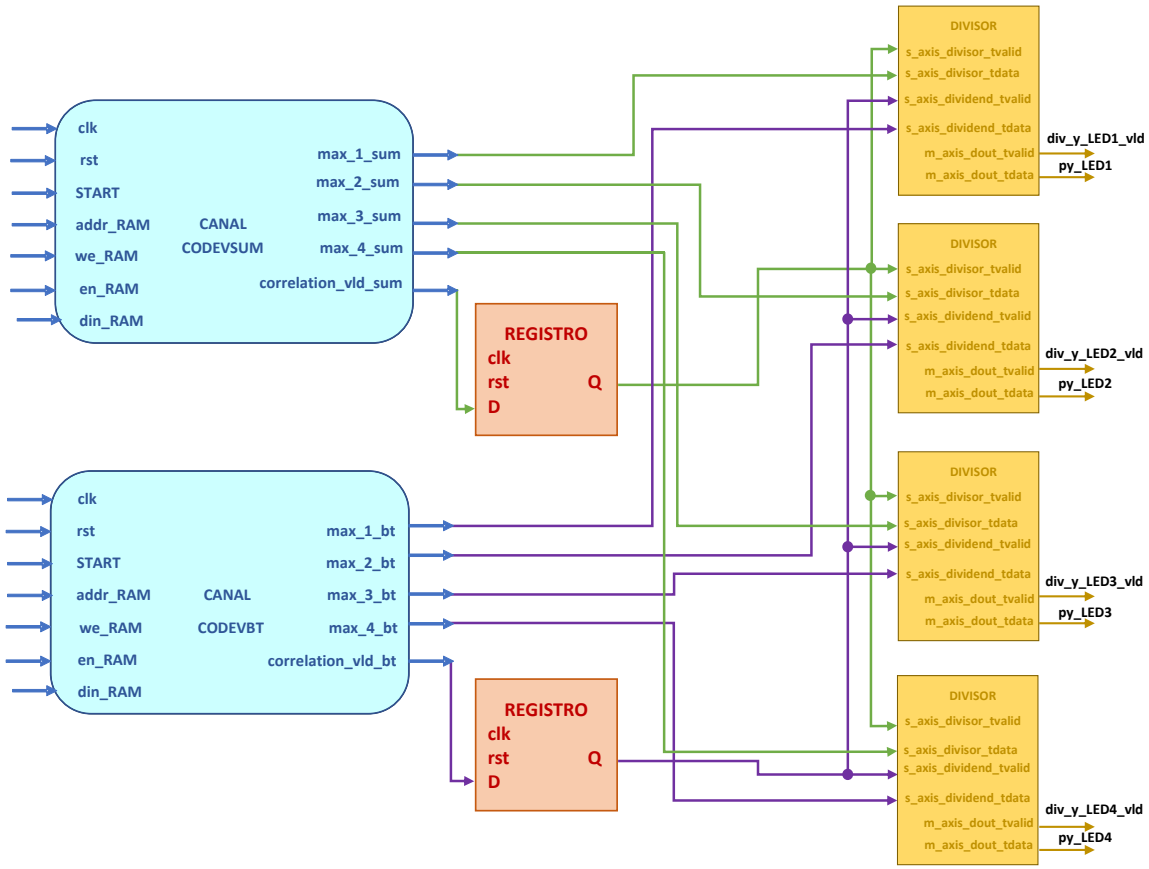

Figura 69. Diagrama de bloques que muestra la conexión entre los módulos canal y los divisores para el cálculo de las relaciones p_x y p_y.

4.2.2. Cálculo del punto de impacto

El último paso para obtener las coordenadas del punto de impacto es multiplicar el resultado de la relación p_x y p_y para cada LED, por la constante $l/2$. Dicha constante fue definida en [1] y la razón de multiplicar por ella se deduce de la Figura 38.

De nuevo se utilizan los cores que ofrece Xilinx VIVADO para realizar la operación de multiplicación, configurándose los ocho cores para multiplicar un valor de 16 bits con signo por otro de 5 bits también con signo, correspondiéndose este último con la constante. Como se vió anteriormente, los multiplicadores utilizados necesitan de una señal de habilitación de reloj *CE*. En este caso, se ha utilizado la señal de validación que genera el módulo divisor cuando termina de operar y muestra el resultado, *m_axis_dout_tvalid*, para habilitar los multiplicadores. En las siguientes figuras se muestra cómo se han realizado las conexiones entre los divisores y los multiplicadores para obtener los valores de las coordenadas buscadas. Concretamente, en la Figura 70 se muestra la arquitectura propuesta para el cálculo de la coordenada *x* del punto de impacto en el receptor para cada uno de los cuatro transmisores LED, mientras que en la Figura 71 se muestra el cálculo de la coordenada *y*, también para todos los transmisores LED.

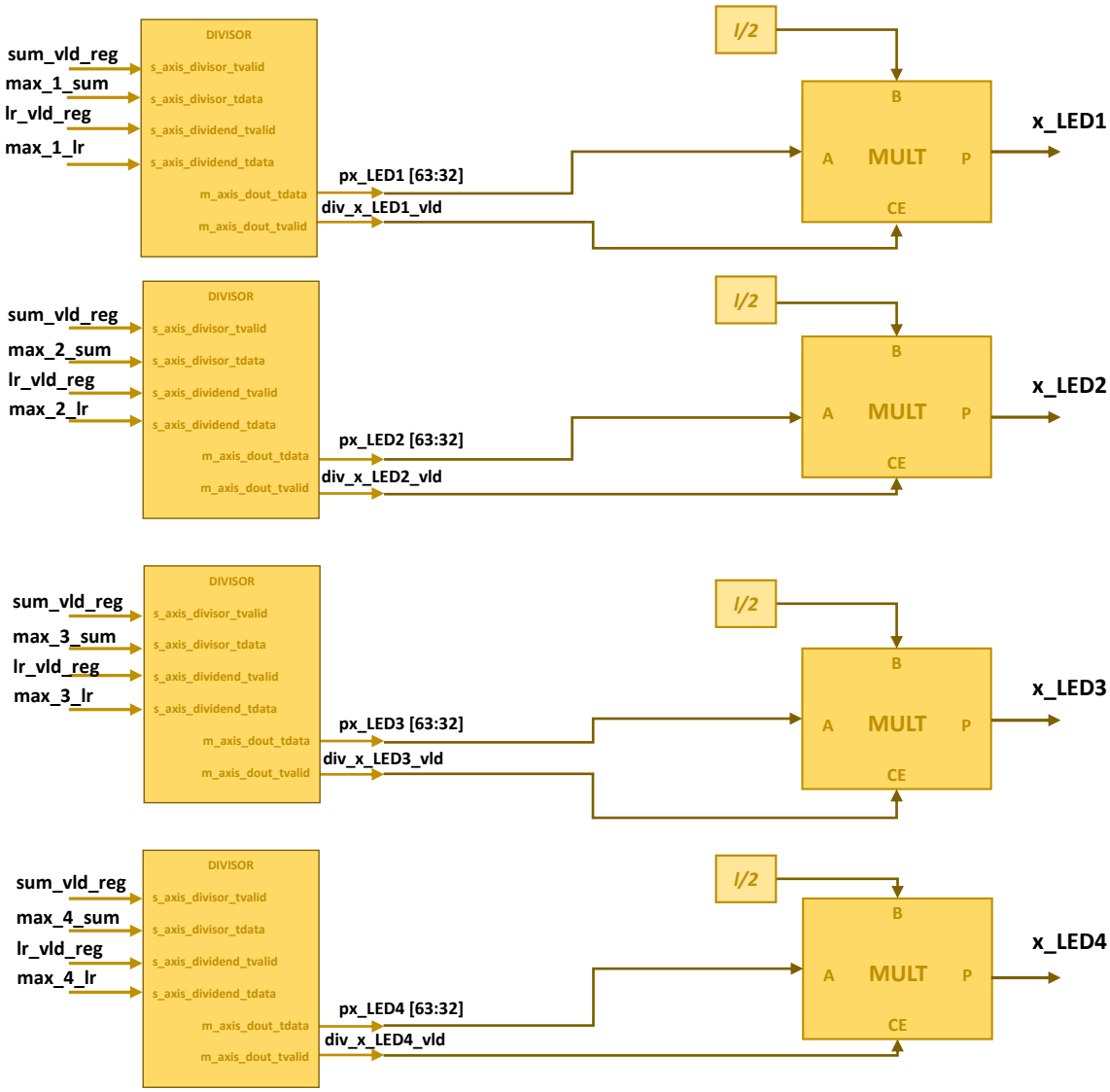

Figura 70. Diagrama de bloques para el cálculo de las coordenadas x para todos los LEDs.

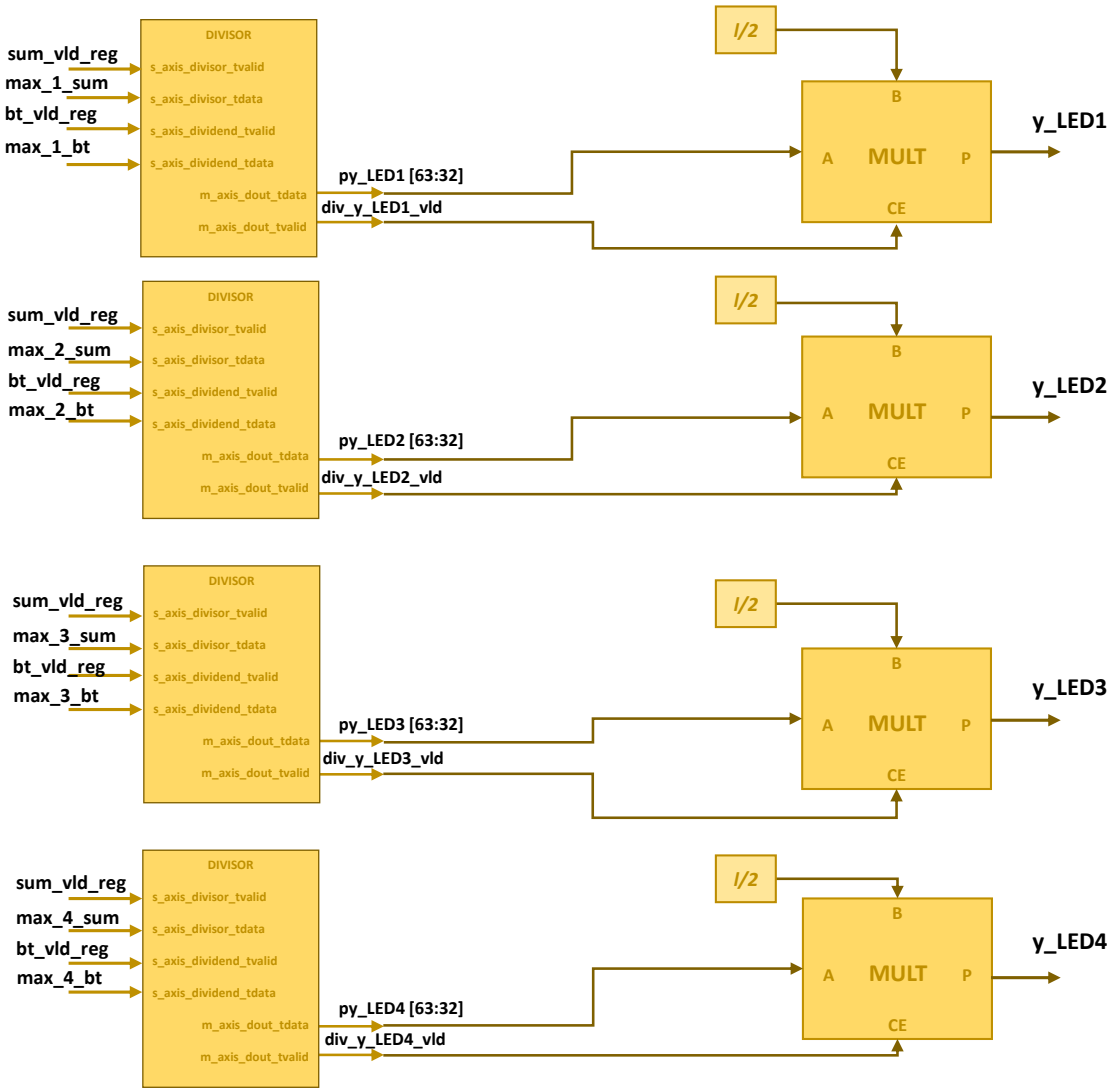

Figura 71. Diagrama de bloques para el cálculo de las coordenadas y para todos los LEDs.

Como pudo observarse en la Figura 66, la arquitectura diseñada también genera una señal de validación que indica que el cálculo de las coordenadas del punto de impacto para cada LED ha finalizado. Esta señal se genera a partir de las señales registradas de validación generadas por los divisores, como se muestra en el esquema completo de la arquitectura propuesta de la Figura 72.

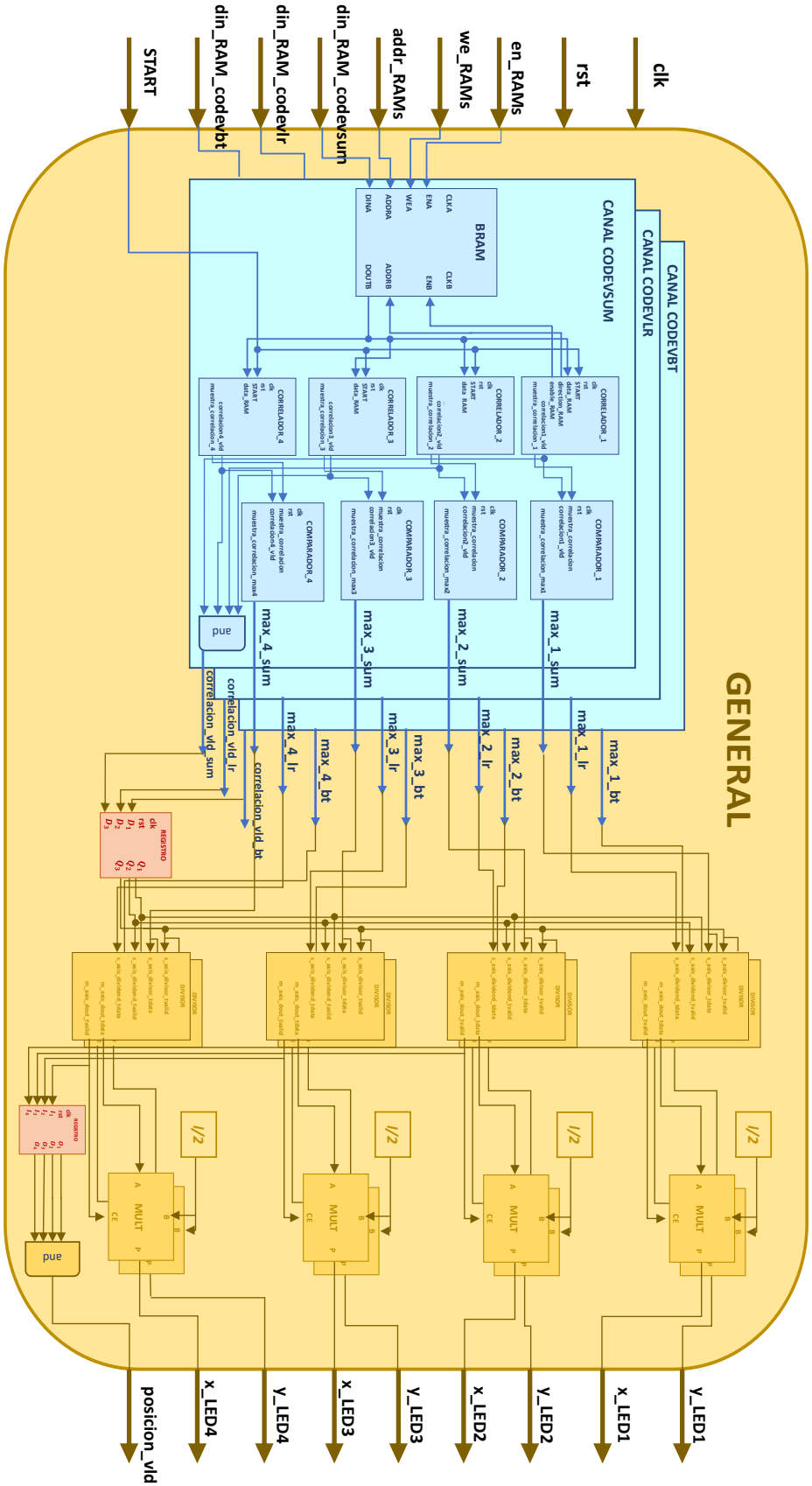

Figura 72. Diagrama de bloques completo de la arquitectura propuesta para el cálculo de las coordenadas de los puntos de impacto.

En la Figura 73, se muestra una simulación funcional de los divisores y multiplicadores que terminan de calcular las coordenadas de los puntos de impacto. Se muestran los puertos de un solo divisor y un multiplicador, pudiéndose extrapolar su comportamiento al resto de los divisores y multiplicadores. Puede observarse que las señales de validación de los operandos, *s_axis_divisor_tvalid* y *s_axis_dividend_tvalid*, son las señales registradas de validación generadas por los módulos *canal*. En este caso, los picos de las correlaciones son iguales para todos los *canales* por lo que el resultado de la división es igual a 1, t = 965 ns. Los puertos *A* y *B* pertenecen a un multiplicador, que opera el resultado de la división con la constante $l/2$. De nuevo por simplificación, se ha fijado el valor de la constante a uno, mostrado en el puerto *B*. Finalmente, en t = 945 ns, se obtienen las coordenadas buscadas.

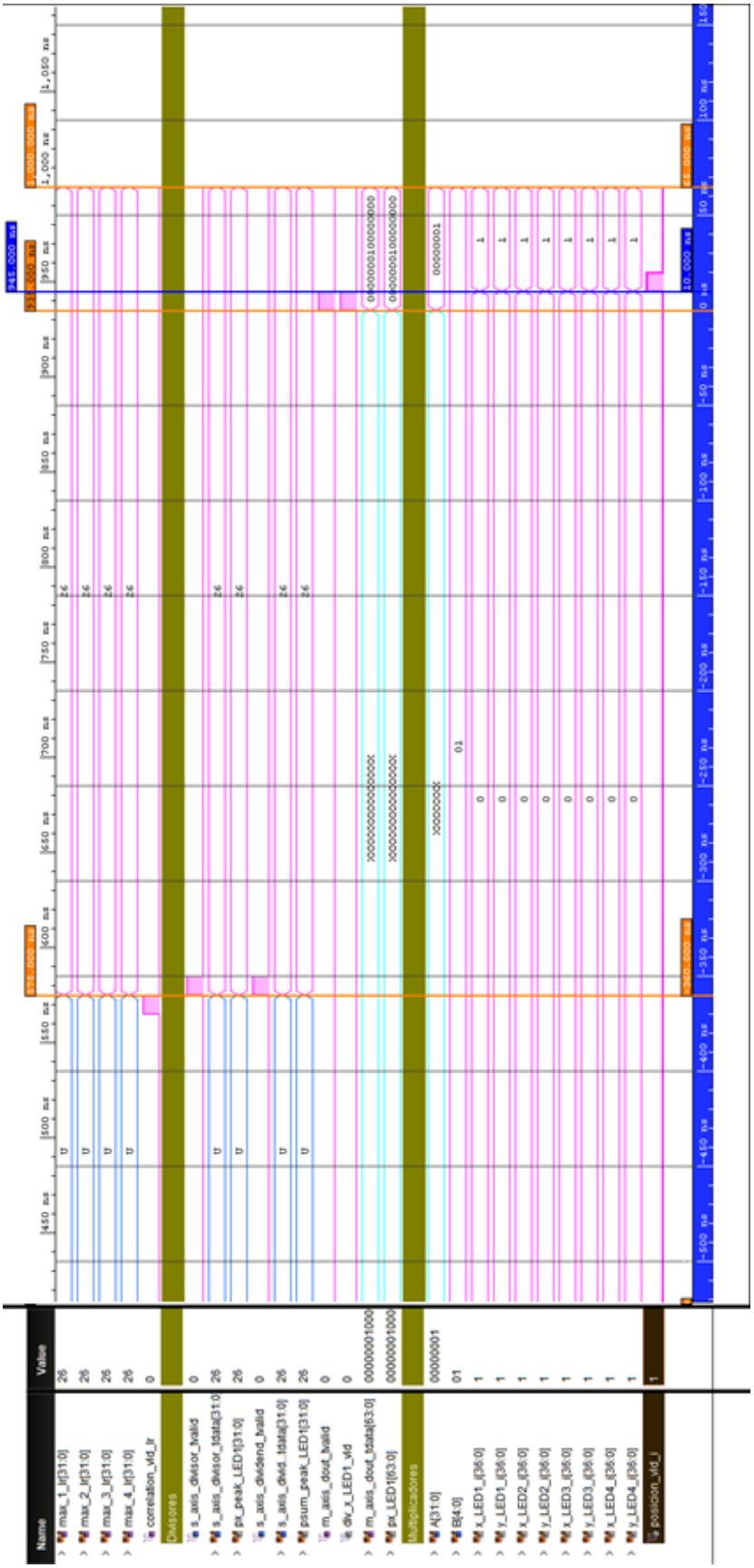

Figura 73. Simulación funcional de los divisores y multiplicadores que realizan el cálculo final de las coordenadas de los puntos de impacto para cada LED.

Capítulo 5

5. Resultados experimentales

Una vez finalizado el diseño de la arquitectura hardware, y comprobado el adecuado funcionamiento de los diferentes módulos que la componen, se procede a realizar simulaciones tanto temporales como funcionales del sistema completo. Se pretende con ello comprobar el correcto funcionamiento del diseño realizado, mediante la comparación de los resultados obtenidos a partir de las simulaciones con los calculados en Matlab.

5.1. Simulación funcional de la arquitectura completa

En el capítulo anterior se mostraron simulaciones funcionales de los módulos individuales que constituyen la arquitectura, inyectando unas señales de entrada al sistema que no se correspondían con las utilizadas para verificar el sistema en [1]. En este punto, se van a mostrar los resultados de la simulación funcional de la arquitectura completa, utilizando las señales experimentales capturadas en escenarios reales.

5.1.1. Adecuación de las señales de entrada

Como se comentó en capítulos anteriores, las señales de salida del receptor QADA (*codevsum*, *codevlr* y *codevbt*) constituyen las muestras de entrada al sistema diseñado. Éstas estaban definidas con un formato en coma flotante y, en el tercer capítulo, hubo que adecuarlas a formato de como fija por ser éste el tipo de dato que una FPGA es capaz de manejar. Tras el análisis, se concluyó que las señales serían representadas con un formato en coma fija Q-6.12, por no cometer un elevado porcentaje de error con respecto a las señales reales, es decir, las representadas en coma flotante. Dicho formato Q-6.12 define que la variable se representa con 12 bits, de los cuales 18 son fraccionarios.

La necesidad de adecuar de nuevo las señales surge de la inviabilidad de introducir como señales de entrada al diseño para su simulación, valores decimales. Con el fin de mantener la coherencia entre los resultados obtenidos mediante Matlab y los que se quieren obtener mediante simulación, se han aplicado una serie de modificaciones a las señales que van a inyectarse durante la simulación. Concretamente, se ha multiplicado cada una de las muestras que constituyen cada una de las señales por el factor 2^{18}, consiguiendo así convertir las muestras decimales en enteras.

Se han generado tres ficheros de texto con las 11540 muestras de cada una de las señales y otros cuatro ficheros *.coe* con cada uno de los cuatro códigos emitidos por los LEDs. Para el caso de los códigos no ha sido necesario adecuar sus valores, pues estos están compuestos por ceros, unos y menos unos.

5.1.2. Análisis de los resultados obtenidos

Para realizar la simulación funcional del sistema con los parámetros experimentales utilizados en [1], se han seguido los siguientes pasos de configuración:

- Se han cargado las doce memorias ROM con los correspondientes ficheros .coe que contienen los códigos emitidos por los LEDs.
- Mediante la especificación de un testbench, que lee las diferentes muestras de las señales contenidas en los ficheros de texto, se han inyectado las señales de entrada (*codevsum*, *codevlr* y *codevbt*) al sistema. Concretamente, se han ido incrementando los buses de direcciones de las tres memorias RAM al tiempo que se han escrito en ellas las muestras de las señales. Una vez guardadas todas las muestras en las memorias, se ha generado un pulso en la señal de START, que desencadena el inicio del cálculo de las coordenadas de impacto en el fotorreceptor QADA.

En la Figura 74 se muestra cómo se ha realizado la escritura de las memorias RAMs, activando la señal *en_RAMs_i* y *we_RAMs_i*, que habilita las memorias y permite la escritura respectivamente, e incrementando posteriormente el bus de direcciones de las memorias RAM sobre la señal *addr_RAMs_i*. Además, se presentan las muestras de las señales *codevsum*, *codevlr* y *codevbt* que se van registrando, en las señales *din_RAM_codevsum*, *din_RAM_codevlr* y *din_RAM_codevbt* respectivamente.

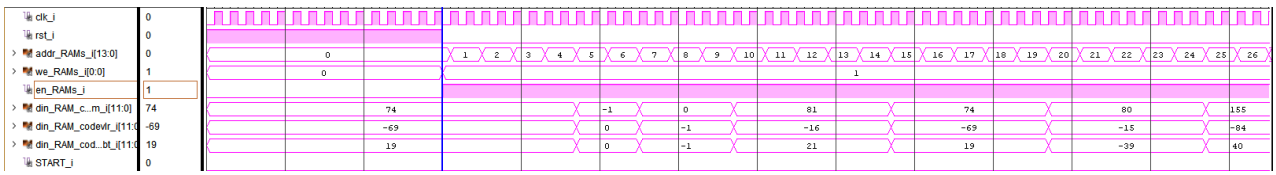

Figura 74. Escritura de las muestras de las señales de entrada al sistema en las memorias RAM.

Sabiendo que las señales contienen 11540 muestras, como se especificó en puntos anteriores, puede intuirse que la señal *addr_RAMs_i* llegará al valor máximo de 11539 y, posteriormente, se generará la señal de START. Tras la activación de dicha señal se inicia el cálculo de las coordenadas. Todo esto puede observarse en la Figura 75 para el canal *Suma* y en la Figura 76 para los canales *BT* y *LR*. Puede comprobarse también en la señal *state* que, al iniciarse el cálculo de las coordenadas, el estado de la máquina de estados diseñada para la gestión de la generación de las direcciones de las memorias, pasa de *reposo* a *first*.

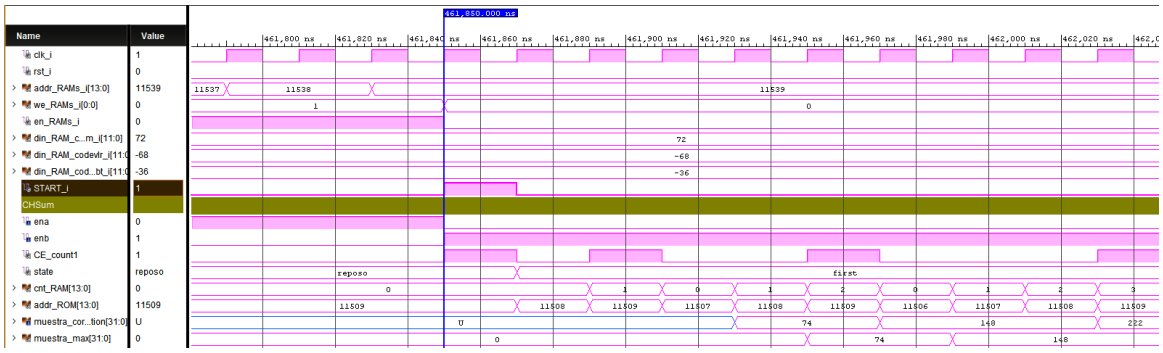

Figura 75. Fin de la escritura de las memorias RAMs e inicio del cálculo de las coordenadas para el canal Sum.

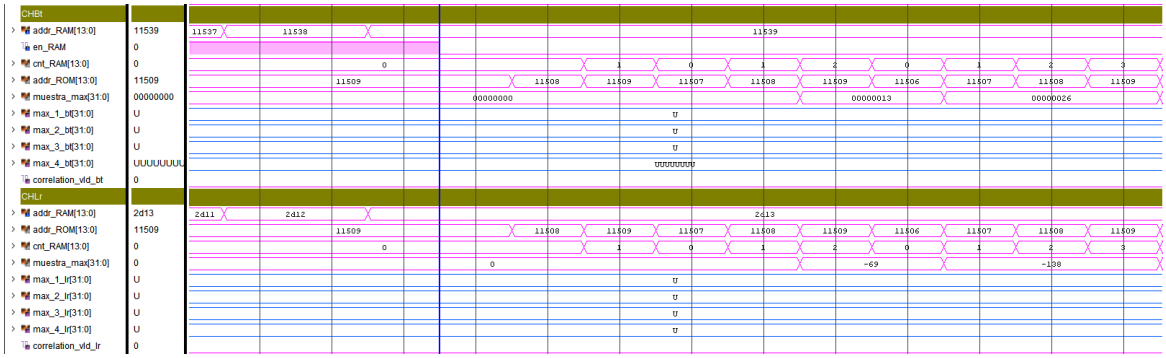

Figura 76. Inicio del cálculo de las coordenadas de los canales BT y LR.

A continuación, se procede a comprobar que los cambios de estado se producen en el momento esperado. Analizando de nuevo la Figura 54 y extrapolando el ejemplo al caso real que en este punto se trata, se obtiene el esquema de la Figura 77. Así, el sistema debe pasar del estado *first* al estado *second* cuando tanto la memoria RAM como la ROM accedan a la posición 11509, que coincide con el número de elementos que constituyen la ROM. Esto se comprueba en la Figura 78, donde *cnt_RAM* y *addr_ROM* representan las direcciones de memoria RAM y ROM a las que se está accediendo respectivamente. Se observa que realmente el sistema cambia de estado una muestra de correlación antes de lo esperado. Esto se debe a que, de esta manera, se permite la actualización de ciertas señales de control del sistema para la adecuada gestión del estado *second*.

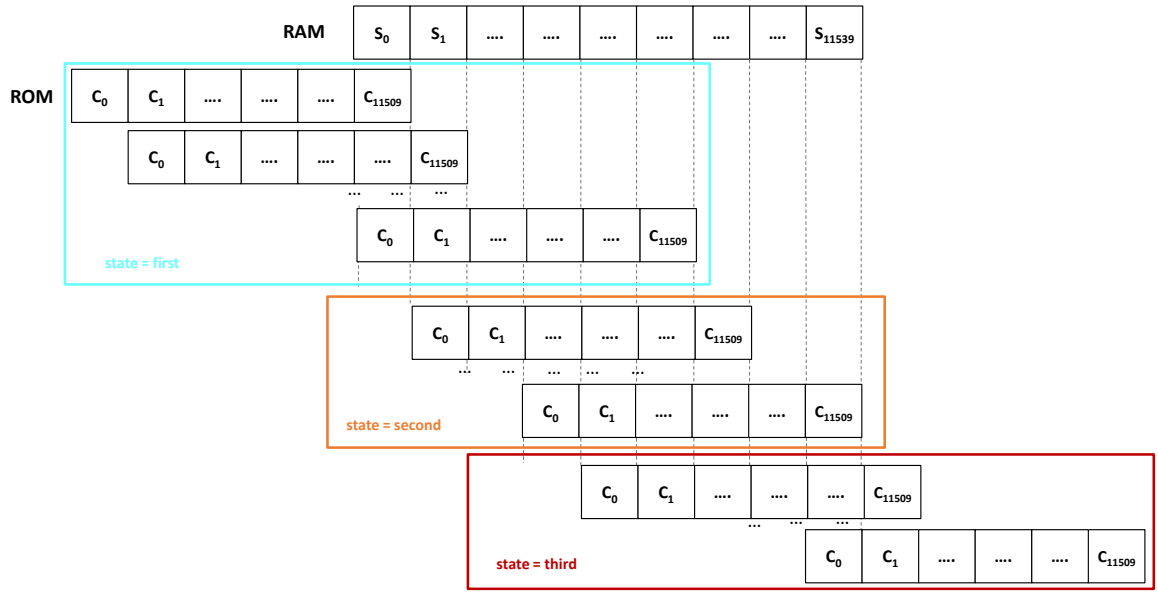

Figura 77. Representación gráfica del proceso de correlación para el caso real.

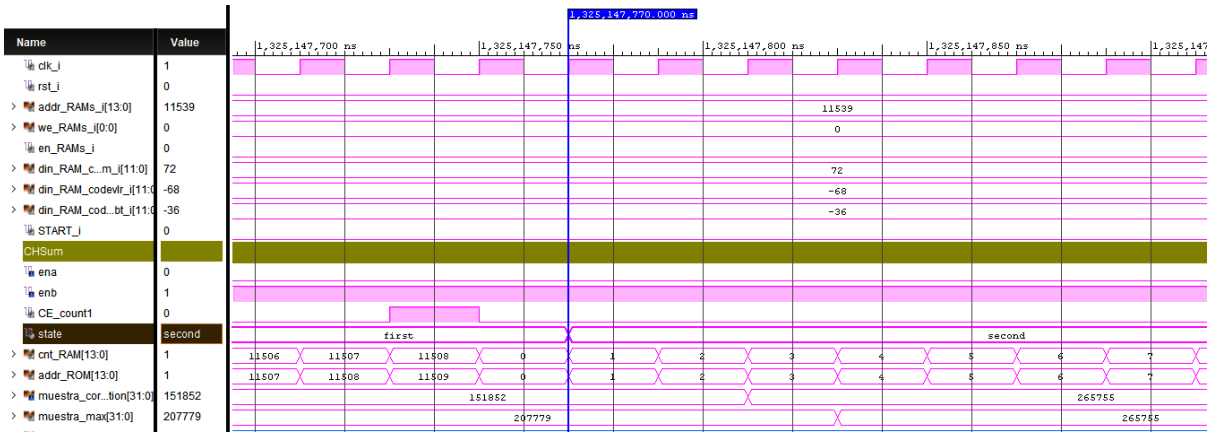

Figura 78. Cambio de estado del sistema, de first a second.

Volviendo a la Figura 77, se deduce que el sistema debe pasar del estado *second* al estado *third* cuando *cnt_RAM* valga 11539 y *addr_ROM* valga 11509; es decir, cuando se esté leyendo simultáneamente la última posición de la memoria RAM y de la memoria ROM. Esto se observa en la Figura 79.

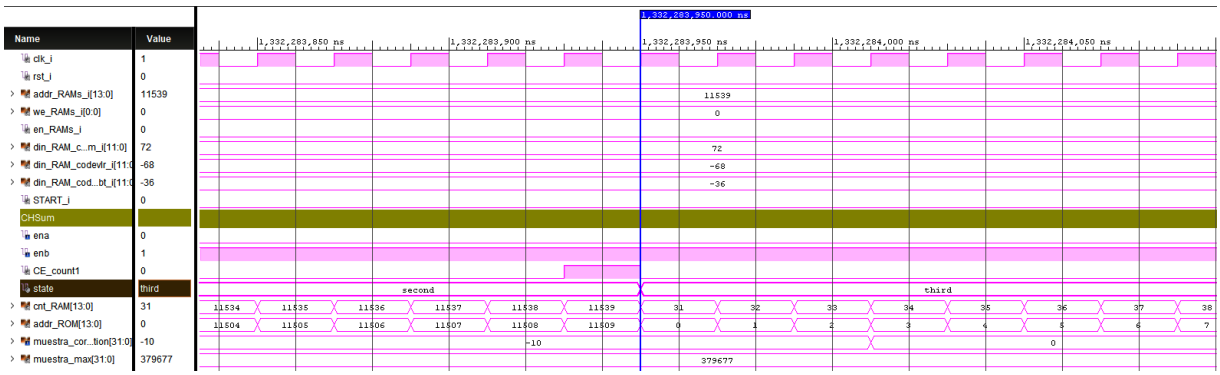

Figura 79. Cambio de estado del sistema, de second a third.

Para comprobar el correcto funcionamiento de los comparadores se muestran, en la Figura 80, algunas muestras de correlación del *canal suma*, calculadas durante el estado *first* y representadas por la señal *muestra_correlation.* La señal que se muestra inmediatamente después, *muestra_max*, es la generada por el comparador. Puede observarse que ésta va actualizando su valor cuando se detecta un máximo absoluto.

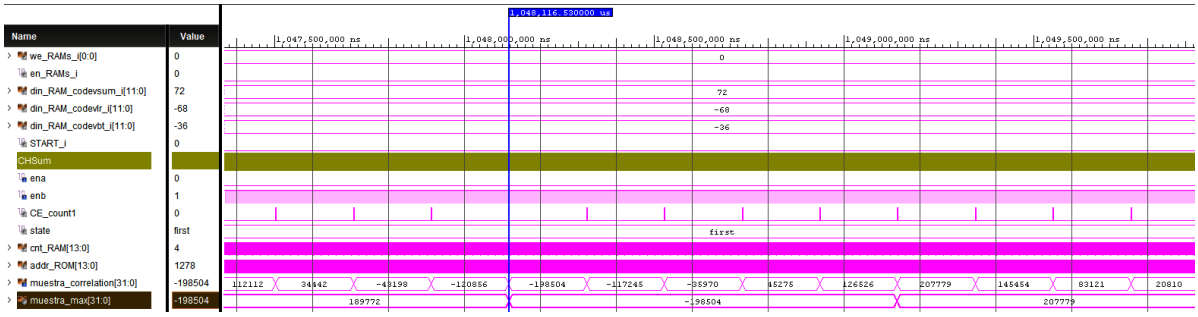

Figura 80. Comprobación del adecuado funcionamiento del comparador.

Por último, queda comprobar que el sistema pasa al estado de *reposo* cuando se leen simultáneamente la última posición de memoria de la RAM, cnt_RAM = 11539, y la primera posición de la memoria ROM, *addr_ROM* = 0. En la Figura 81, se muestra la llegada del sistema al estado *reposo* para el canal Sum, BT y LR.

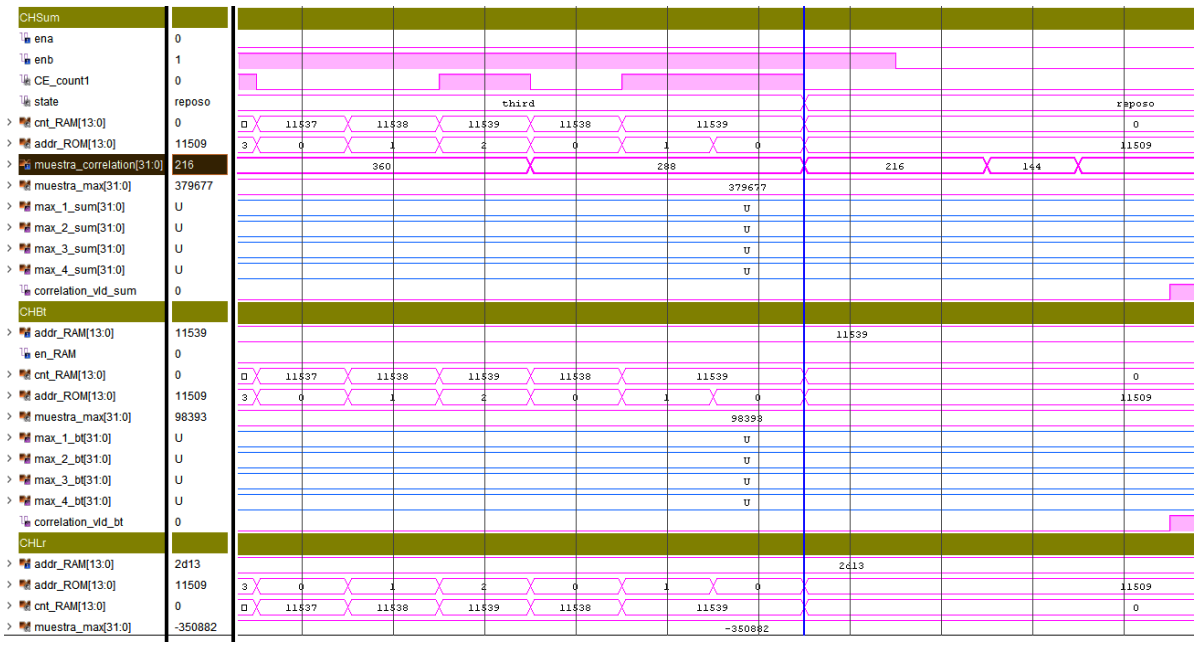

Figura 81. Transición de los canales Sum, BT y LR al estado reposo.

Cuando el sistema llega al estado *reposo*, las correlaciones ya han sido calculadas, junto con los valores máximos de las mismas. En la Figura 82, se muestra el resultado de correlar la señal *codevsum* con el código emitido por el LED1 junto con los máximos hallados, representados por la señal *muestra_max*. En la Figura 83, se muestra el resultado de la misma correlación pero calculada y cuantificada en Matlab. Esto se realizó durante el estudio previo de adecuación de las señales del sistema en el capítulo 3. Comparando estas dos figuras, se observa que ambas tienen una envolvente similar y, además, el pico máximo de correlación coincide. Para la

correlación calculada en Matlab, se muestra que dicho pico es igual a 1,4465, mientras que para la calculada por la arquitectura diseñada el valor máximo es 379677. Si a dicho máximo se le realiza la operación inversa, explicado en el punto anterior, se obtiene aproximadamente el mismo valor: $379677 \cdot 2^{-18} = 1.448$.

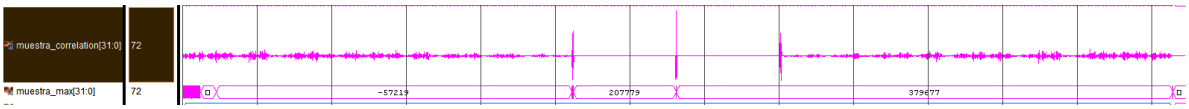

Figura 82. Señal de correlación resultado de correlar la señal codevsum con el código emitido por el LED1.

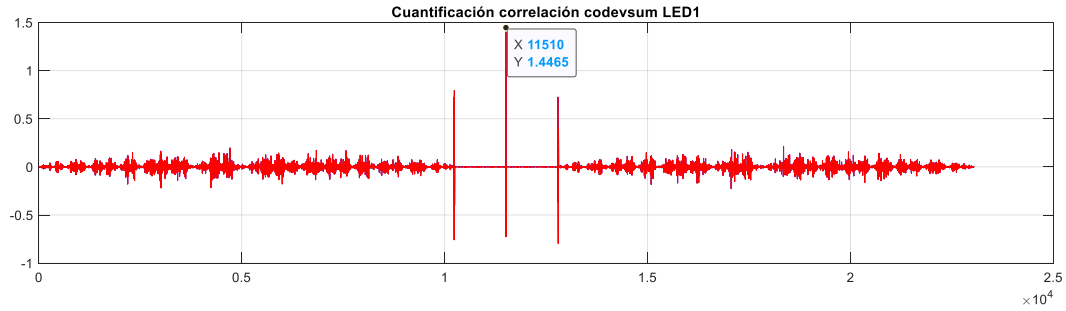

Figura 83. Resultado de la correlación de la señal codevsum con el código emitido por el LED1 cuantificada en Matlab.

A continuación, en la Figura 84, Figura 85 y Figura 86, se muestran los resultados máximos de las correlaciones realizadas para el canal *Sum*, *BT* y *LR* respectivamente.

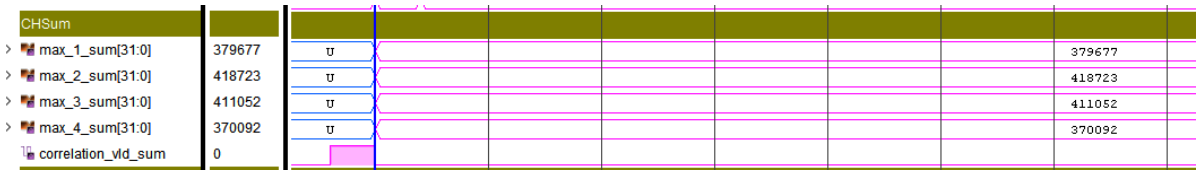

Figura 84. Valores máximos de correlación del canal suma.

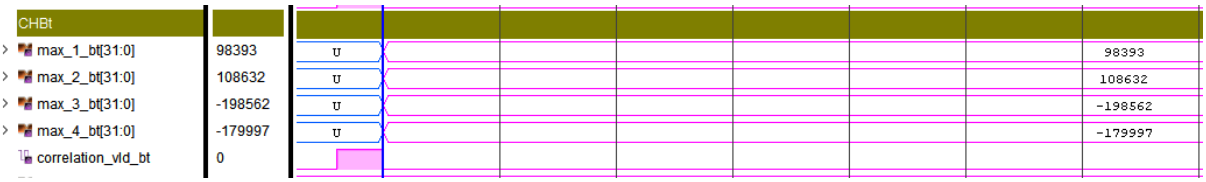

Figura 85. Valores máximos de correlación del canal BT.

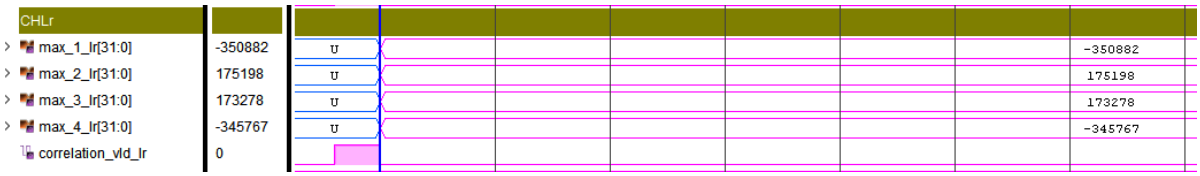

Figura 86. Valores máximos de correlación del canal LR.

Una vez calculados los máximos de las correlaciones, el sistema pasa a calcular las relaciones p_x y p_y para cada LED. En la Figura 87, se muestra el cálculo de la relación p_x del LED1 mediante la división del valor máximo de la correlación del LED1 del canal *codevlr* con el valor máximo de

correlación del LED1 del canal *codevsum*. En la Figura 88, se muestra el cálculo de la relación p_y del LED1 mediante la división del valor máximo de correlación del LED1 del canal *codevbt* con el valor máximo de correlación del LED1 del canal *codevsum*. Los divisores de la arquitectura realizan divisiones enteras, por lo que al ser el resultado de las mimas inferior a uno, el cociente que queda es igual a cero. Esto se refleja en las señales *px_LED1* y *py_LED1*, que se corresponden con las salidas de los divisores. Éste entrega un dato de 64 bits, de los cuales los 32 de mayor peso se corresponden con el cociente de la división y los bits restantes con el resto.

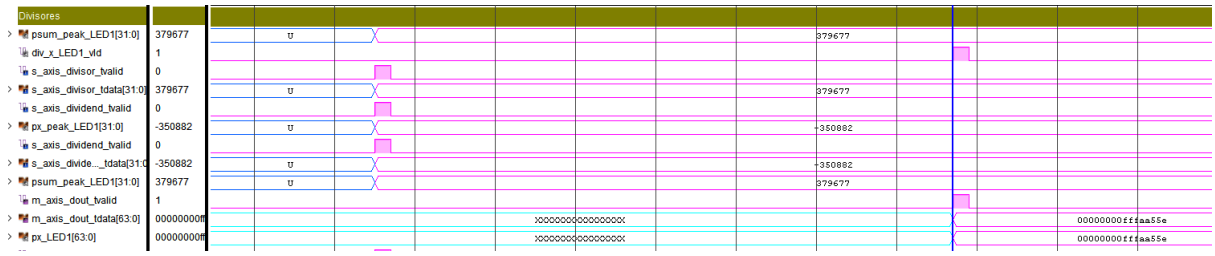

Figura 87. Cálculo de la relación p_x del LED1.

Figura 88. Cálculo de la relación p_y del LED1.

5.2. Simulación temporal de la arquitectura completa

Para la realización de las simulaciones funcionales, el simulador integrado en el entorno Vivado y utilizado en este TFG, emplea modelos de diseño descritos en HDL. Esto hace que este tipo de simulaciones sean más rápidas, por no contener dichos modelos ningún tipo de información temporal. Son muy útiles, pues permiten verificar la funcionalidad del diseño desarrollado en un corto periodo de tiempo, pero no permiten verificar que efectivamente la arquitectura funcione en el dispositivo de destino. Para esto es necesario realizar una simulación temporal.

El primer paso para realizar una simulación temporal completa es la síntesis del código. En este proceso, las especificaciones del diseño desarrollado en alto nivel de abstracción se convierten en representaciones optimizadas a nivel de puertas (*netlist*). Después, se realiza el proceso de implementación, donde la herramienta Vivado ubica el diseño en los diferentes elementos de la FPGA. Tras esto, el simulador tiene información acerca de los retardos estimados de los modelos de los dispositivos y los originados por las interconexiones. Por lo tanto, con esta simulación puede resolverse si el diseño realizado es capaz de funcionar correctamente y a la velocidad establecida. Se trata de una simulación lenta pero necesaria para la verificación final del diseño.

Para simplificar el proceso de análisis de la simulación temporal y reducir el tiempo de espera, de nuevo se ha optado por introducir al diseño unas señales simplificadas. La escritura de las memorias RAM se ha llevado a cabo de la misma manera que en la simulación funcional. En Figura 89 se muestra el proceso.

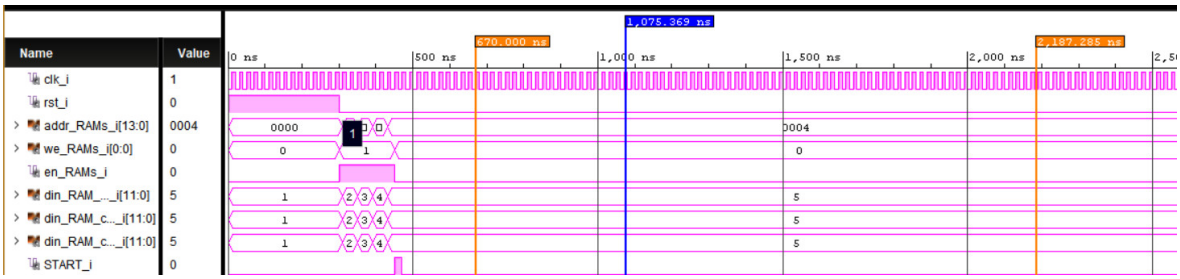

Figura 89. Escritura de las muestras de las señales de entrada al sistema en las memorias RAM.

En la Figura 90, Figura 91 y Figura 92, se muestran los valores máximos absolutos obtenidos tras realizar las correlaciones para los canales *suma*, *BT* y *LR* respectivamente. Se observan fallos evidentes, pues habiendo sido cargadas las memorias RAM con los mismos valores, todos los máximos absolutos obtenidos deberían ser los mismos, 26. Esto sólo se da, para las cuatro correlaciones del canal *LR*.

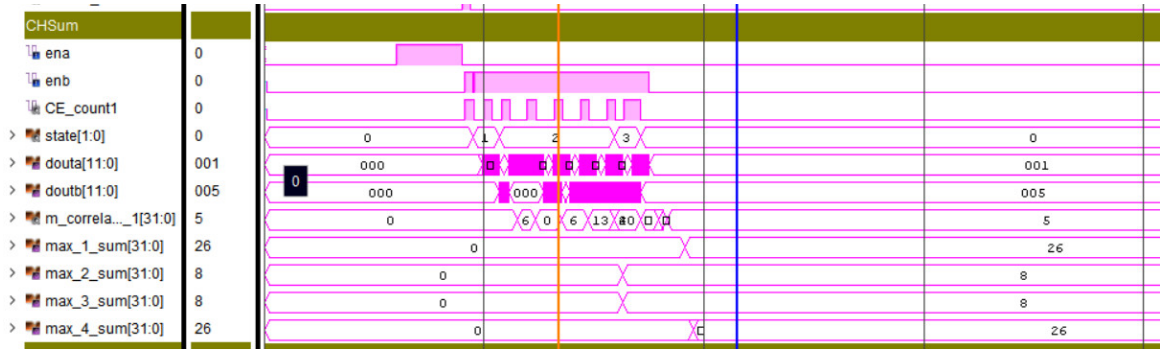

Figura 90. Cálculo de las correlaciones y búsqueda de picos máximos del canal suma.

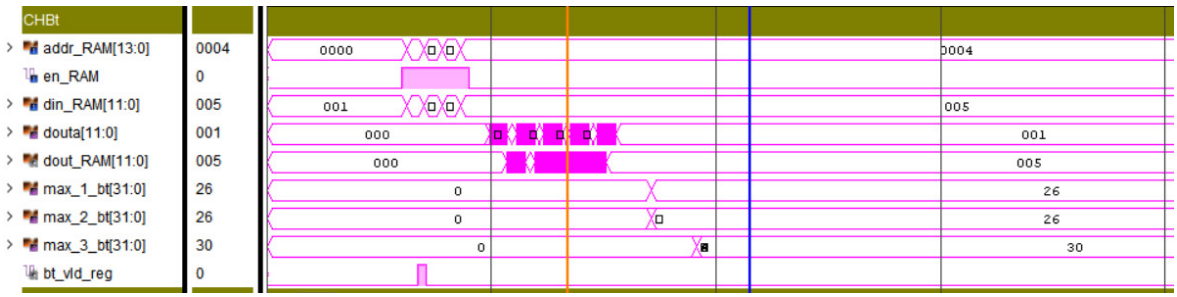

Figura 91. Cálculo de las correlaciones y búsqueda de picos máximos del canal Bottom-Top.

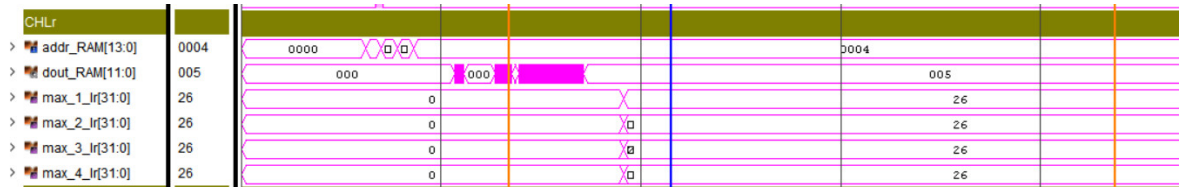

Figura 92. Cálculo de las correlaciones y búsqueda de picos máximos del canal Left-Right.

Además, en el cálculo de las relaciones p_x y p_y también se ha detectado un fallo. Se observa en la Figura 93, como se generan dos pulsos que validan el valor del divisor y dividendo, cuando todavía no han sido calculados. Asimismo, cuando éstos ya han sido calculados no se genera el pulso de validación del divisor. Por esto y por lo comentado anteriormente, como puede observarse en la Figura 94, el cálculo de las coordenadas de impacto es erróneo.

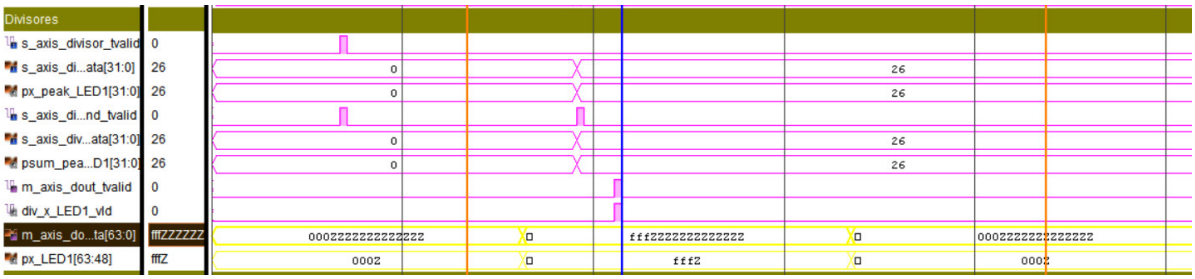

Figura 93. Cálculo de la relación p_x para el LED1.

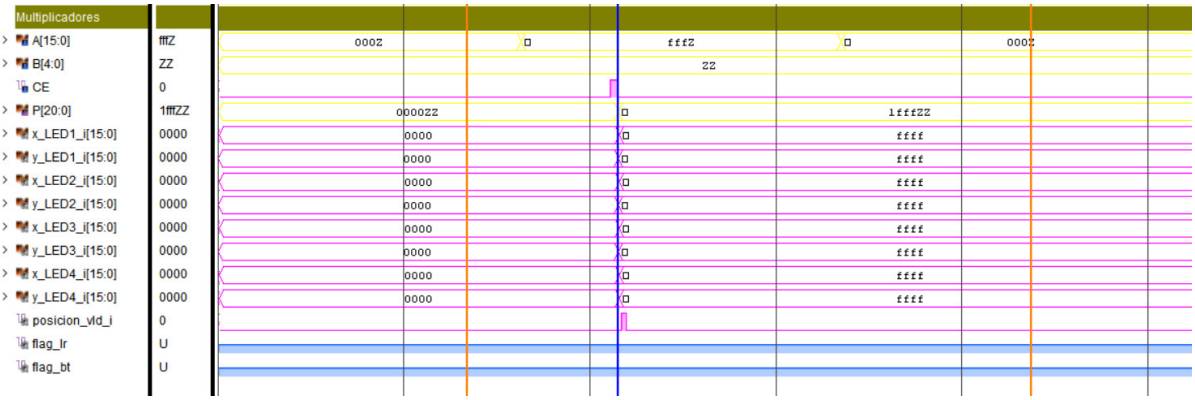

Figura 94. Valores de las coordenadas de los puntos de impacto en el receptor QADA para cada LED.

Es necesario por tanto seguir depurando el diseño, teniendo en cuenta que esta simulación temporal necesita para ejecutarse por completo alrededor de las 3 horas.

5.3. Consumo de recursos

Los recursos de la FPGA utilizados para la implementación del algoritmo de posicionamiento pueden obtenerse después de la síntesis del código o después de la implementación. Una vez realizada la síntesis, los datos que se pueden extraer son una estimación de los recursos a utilizar. Sin embargo, durante la implementación Vivado realiza una optimización de recursos, emplaza el diseño y fija las interconexiones, por lo que tras este proceso la información del consumo de recursos es final. Por ello, todos los datos que se muestran a continuación han sido extraídos tras la implementación.

En la Tabla 11, Tabla 12 y Tabla 13 se muestran los recursos utilizados para el canal suma, BT y LR respectivamente. En ellas se observa que los correladores son, con diferencia, los módulos que más recursos consumen dentro de los bloques *canal*.

	LUT	FF	BRAM
CHANELSUM	1952	836	27.5
RAM	23	4	5.5
CORRELADOR1	419	144	5.5
CORRELADOR2	419	144	5.5
CORRELADOR3	419	144	5.5
CORRELADOR4	419	419	5.5
COMPARADOR1	63	64	0
COMPARADOR2	63	64	0
COMPARADOR3	63	64	0
COMPARADOR4	63	64	0

Tabla 11. Recursos de la FPGA utilizados para implementar el canal suma.

	LUT	FF	BRAM
CHANELBT	1522	663	22
RAM	23	4	5.5
CORRELADOR1	419	144	5.5
CORRELADOR2	419	144	5.5
CORRELADOR3	419	144	5.5
CORRELADOR4	52	35	0
COMPARADOR1	63	64	0
COMPARADOR2	63	64	0
COMPARADOR3	63	64	0
COMPARADOR4	63	64	0

Tabla 12. Recursos de la FPGA utilizados para implementar el canal BT.

	LUT	FF	BRAM
CHANELLR	1952	836	27.5
RAM	23	4	5.5
CORRELADOR1	419	144	5.5
CORRELADOR2	419	144	5.5
CORRELADOR3	419	144	5.5
CORRELADOR4	419	144	5.5
COMPARADOR1	63	64	0
COMPARADOR2	63	64	0
COMPARADOR3	63	64	0
COMPARADOR4	64	64	0

Tabla 13. Recursos de la FPGA utilizados para implementar el canal LR.

Sin embargo, son los divisores que se encargan de calcular las relaciones p_x y p_y los que más recursos consumen del diseño completo. Esto se observa en la Tabla 14.

Name	Slice LUTs (53200)	Slice Registers (106400)	Block RAM Tile (140)
MultXLED2 (mult_X_L...	2	16	0
MultXLED1 (mult_X_L...	2	16	0
DivPyLED4 (div_Y_LE...	1217	3189	0
DivPyLED3 (div_Y_LE...	1217	3189	0
DivPyLED2 (div_Y_LE...	1216	3189	0
DivPyLED1 (div_Y_LE...	1217	3189	0
DivPxLED4 (div_X_LE...	1217	3189	0
DivPxLED3 (div_X_LE...	1217	3189	0
DivPxLED2 (div_X_LE...	1217	3189	0
DivPxLED1 (div_X_LE...	1217	3189	0

Tabla 14. Utilización de los recursos de la FPGA de los divisores.

Por último, se muestra un resumen del porcentaje de recursos utilizados por cada bloque que compone la arquitectura, así como el total utilizado por el diseño completo.

	%LUTS	%FF	%BRAM
CHANELSUM	3.67%	0.78%	19.64%
CHANELBT	2.86%	0.62%	15.71%
CHANELLR	3.67%	0.78%	19.64%
MUTIPLICADORES	0.03%	0.12%	0%
DIVISORES	18.3%	7%	0%
ARQUITECTURA COMPLETA	28.53%	9.3%	54.99%

Tabla 15. Porcentaje de ocupación de recursos de la FPGA de la arquitectura.

5.4. Frecuencia máxima de funcionamiento

Antes de realizar la implementación del diseño ha de fijarse una restricción temporal, indicando cuál es la frecuencia de funcionamiento que ha de tener el sistema. Tras la implementación, Vivado indicará si el diseño puede funcionar correctamente a esa frecuencia mostrando un resumen de temporización.

El desarrollo del diseño se inició fijando una frecuencia de funcionamiento de 100 MHz, de hecho, algunas de las simulaciones funcionales mostradas trabajan a esta frecuencia. Al realizar la implementación se obtuvo el timing del sistema, mostrado en la Figura 96.

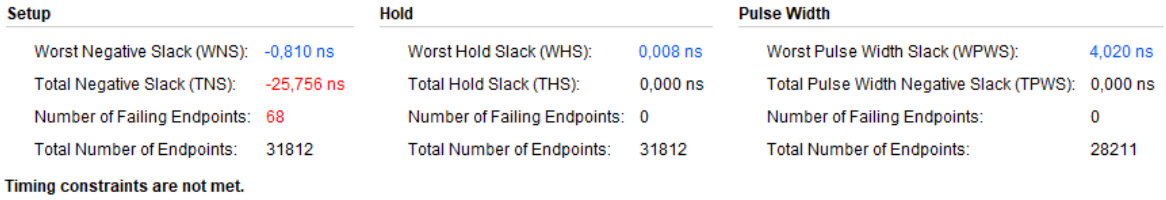

Figura 95. Resumen de temporización para una frecuencia de 100 MHz.

Tras observar que la temporización fallaba en el tiempo de setup, concretamente en 68 rutas, se disminuyó la frecuencia de reloj a la mitad y se volvió a realizar la implementación. En este caso, los resultados obtenidos se reflejan en la Figura 96. Se observa que para dicha frecuencia los resultados son positivos.

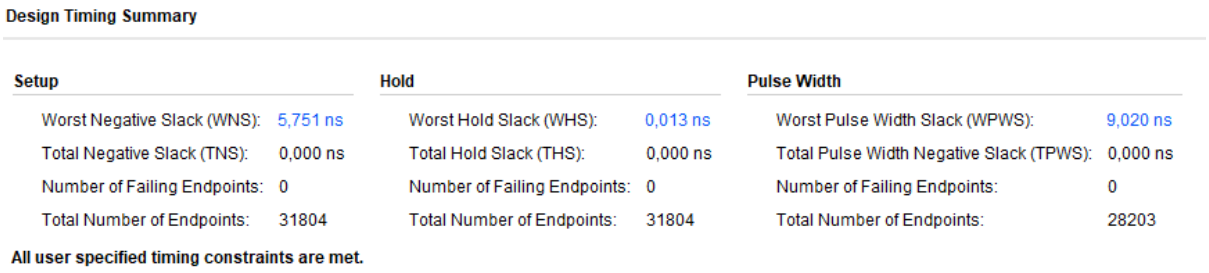

Figura 96. Resumen de temporización para una frecuencia de 50 MHz.

Capítulo 6

6. Conclusiones y trabajos futuros

Tras el análisis del sistema de posicionamiento descrito en [1], la comprensión de la problemática a resolver mediante el diseño de una arquitectura en un dispositivo FPGA y la propuesta final de dicha arquitectura, junto con la verificación funcional de la misma, el desarrollo del presente trabajo de fin de grado se considera concluido. En los siguientes puntos se resumen las conclusiones finales, así como las posibles mejoras y ampliaciones a desarrollar en un futuro.

6.1. Conclusiones

En este trabajo, se ha presentado una arquitectura VHDL del sistema de posicionamiento óptico basado en técnicas de codificación desarrollado en [1]. Dicha arquitectura es capaz de, a partir de tres señales de entrada entregadas por un fotorreceptor QADA, calcular las coordenadas de los puntos de impacto sobre el QADA para cada transmisor LED. Esto es posible gracias a la correlación de las señales de entrada con los códigos emitidos por cada transmisor LED, a la posterior búsqueda de los valores máximos absolutos de los resultados de dicha operación y a la realización de una serie de consideraciones geométricas sobre los valores máximos. A partir de esta conclusión principal se deducen las siguientes conclusiones parciales:

- Se ha realizado un análisis del algoritmo de posicionamiento modelado en Matlab y una posterior adecuación del mismo, realizando la conversión de las diferentes señales implicadas, de formato en coma flotante a formato en coma fija, buscando siempre la menor pérdida de información. A partir de este estudio, se ha podido fijar el número de bits necesarios para la adecuada representación de las señales en cada etapa del algoritmo. Se ha conseguido con ello una utilización eficiente de los recursos de la FPGA.
- Para el cálculo de las correlaciones y la búsqueda de los valores máximos absolutos de los resultados, se ha desarrollado una arquitectura, denominada *canal*, que contiene los módulos capaces de realizar dichas operaciones:
    - El módulo *correlador*, que se encarga de correlar una de las señales de salida del fotorreceptor QADA con uno de los códigos emitidos por los LEDs.
    - El módulo *comparador* que, mediante la comparación de las diferentes muestras de correlación generadas por un módulo *correlador,* es capaz de

determinar cuál es el valor máximo absoluto de entre todas las muestras analizadas.
    - Una máquina de estados, que se encarga de sincronizar y gestionar el correcto funcionamiento de los módulos *comparador* y *correlador*.
- Se ha desarrollado un bloque superior, denominado *general*, que constituye la arquitectura del sistema completo. Este bloque contiene tres entidades *canal*, cada una de ellas orientadas al tratamiento de una de las tres señales de salida del QADA, y una serie de multiplicadores y divisores. Dicho bloque es capaz de, a partir de los valores máximos absolutos de las correlaciones, calcular las buscadas coordenadas de los puntos de impacto de cada LED sobre el receptor QADA.

6.2. Trabajos futuros

En relación con los resultados obtenidos de la arquitectura propuesta y el análisis objetivo de la misma, se plantean los posibles trabajos futuros a realizar:

- Es posible mejorar la eficiencia del sistema, disminuyendo la utilización de recursos de la FPGA. Como se comentó anteriormente, cada uno de los cuatro módulos *correlador* presentes en las tres entidades *canal*, contienen una memoria ROM con las muestras correspondientes a uno de los cuatro códigos emitidos por cada LED. En total, la arquitectura diseñada tiene 12 memorias ROM, conteniendo cada una de ellas la misma información que otras dos memorias. Se propone por tanto como trabajo futuro eliminar las memorias ROM de los módulos *correlador* e instanciar cuatro memorias ROM en la entidad superior *general*. Cada memoria contendrá uno de los códigos emitidos por cada uno de los cuatro transmisores LED y se dotará al módulo *correlador* con la capacidad de acceder a los datos de las memorias situadas en una entidad externa. Cabe señalar que esta modificación es viable siempre y cuando todas las señales de entrada al sistema estén definidas por el mismo número de muestras. Se propone, también, refinar el diseño completo en busca de situaciones similares a la anteriormente comentada.
- Como se comentó anteriormente, no se ha refinado lo suficiente el código por lo que, actualmente, el diseño no funciona en la FPGA. Se propone por lo tanto, la depuración exhaustiva del código para conseguir una adecuada simulación temporal y con ello, un correcto funcionamiento en la FPGA.
- En este trabajo se ha propuesto un diseño de una arquitectura capaz de realizar paralelamente la correlación de señales, deducir los máximos absolutos de los resultados y calcular con ellos las coordenadas de los puntos de impacto en el fotorreceptor QADA para cada transmisor LED. El sistema desarrollado en [1] es capaz de, a partir de dichas coordenadas, calcular la posición del receptor respecto a una celda unitaria de 2x2x2 cm. Se propone entonces como trabajo futuro la implementación del algoritmo que genera las coordenadas finales del receptor. Se plantea, además, que dicha implementación, se realice en la parte de sistema de procesamiento de la FPGA debido a la alta carga computacional requerida por dicho algoritmo.
- Se propone desarrollar la arquitectura diseñada mediante el lenguaje de alto nivel HLS, para evaluar con qué herramienta se consigue un sistema más eficiente.

Pliego de condiciones

A continuación, se especifican los recursos hardware y software utilizados durante el desarrollo de este trabajo de fin de grado.

Recursos hardware:

- Ordenador portátil MacBook Pro con procesador Intel Core i5 dual core de 2.4GHz, 4GB de memoria integrada DDR3L de 1600MHz.
- Disco externo de estado sólido SSD de 1TB.
- Plataforma Zedboard, que incluye una FPGA de la familiar Zynq de Xilinx, Inc.

Recursos software:

- Sistemas operativos:
    - Windows 10 pro
    - MacOS Catalina
- Matlab © R2019a
- Vivado 2017.4
- Editor de texto Word
- PowerPoint
- Gimp 2

Presupuesto

Los recursos hardware y software utilizados, llevan asociados unos costes que se especifican en la Tabla 16 y Tabla 17, respectivamente.

- **HARDWARE UTILIZADO**

HARDWARE	PRECIO	DURACIÓN	USO	TOTAL
MacBook Pro	1500€	7 años	6 meses	107.14€
SSD externo	130€	3 años	6 meses	21.66€
ZedBoard	485€	-	-	485€
Subtotal				**613.8€**

Tabla 16. Presupuesto de los recursos hardware utilizados.

- **SOFTWARE UTILIZADO**

SOFTWARE	PRECIO	DURACIÓN	USO	TOTAL
Windows 10 pro	20€	3 años	6 meses	3.33€
MAC	0€	2 años	6 meses	0€
Paquete Office: - Word - PowerPoint	0€	3 años	6 meses	0€
Matlab © R2019a	0€	3 años	6 meses	0€
Gimp2	0€	2 meses	6 meses	0€
Subtotal				**3.33€**

Tabla 17. Presupuesto de los recursos software utilizados.

- **MANO DE OBRA**

Existe un coste añadido derivado de la mano de obra necesaria para la realización del proyecto. Se ha fijado para su cálculo, un salario medio de un ingeniero graduado y se han contabilizado

el total de horas empleadas, tanto para el desarrollo técnico del trabajo como para su descripción en un documento formal.

TRABAJADOR	Nº DE HORAS	€/HORA	TOTAL
Graduado	580h	35€	20300€
SUBTOTAL			**20300€**

Tabla 18. Presupuesto de la mano de obra.

- **COSTE TOTAL**

Finalmente, se calcula en la Tabla 19 el presupuesto total del proyecto. Para ello se suman los costes por los recursos software y hardware, la mano de obra, los gastos generales, asociados al gasto generado por el uso de instalaciones y, por último, se incluye el impuesto sobre el valor añadido (IVA).

DESCRIPCIÓN	COSTE SIN IVA	IVA	TOTAL
Recursos hardware	613.8€	21%	742.7€
Recursos software	3.33€	21%	4€
Mano de obra	20300€	21%	24563€
Infraestructuras	60€	21%	72.6€
SUBTOTAL			**25382.3€**

Tabla 19. Presupuesto total del TFG.

El presupuesto total del proyecto es de VEINTICINCO MIL TRESCIENTOS OCHENTA Y DOS EUROS CON TREINTA CÉNTIMOS DE EURO.

Bibliografía

[1] E. Aparicio-Esteve. “Design and implementation of an optical positioning system based on encoding techniques”. Máster Thesis, Departamento de electrónica, Universidad de Alcalá de Henares. Madrid. 2019.

[2] Oficina de Coordinación Nacional de Posicionamiento, Navegación y Cronometría por Satélite del gobierno de los Estados Unidos. ”El sistema de posicinamiento Global”. [En línea]. Disponible en: https://www.gps.gov/spanish.php. [Accecido: 26 de mayo de 2021]

[3] C. Hackman, S. M. Byram, V. J. Slabinski and J. C. Tracey, "USNO GPS/GLONASS PNT products: Overview, and GPS+GLONASS vs GLONASS only PPP accuracy," 2014 IEEE/ION Position, Location and Navigation Symposium - PLANS 2014, Monterey, CA, 2014, pp. 793-803

[4] GlobalSecurity.org. “Beidou (Big Dipper / COMPASS)”. [En Línea]. Disponible en: https://www.globalsecurity.org/space/world/china/beidou.htm. [Consultado: 26 de mayo de 2021].

[5] “What is Galileo?”, Archive.org. [En línea]. Disponible en: https://web.archive.org/web/20190714102703/https://www.gsc-europa.eu/galileo-overview/what-is-galileo. [Consultado: 26-mayo-2021].

[6] P. Pascacio, S. Casteleyn, J. Torres-Sospedra, E. S. Lohan, y J. Nurmi, “Collaborative Indoor Positioning Systems: A systematic review”, Sensors (Basel), vol. 21, núm. 3, p. 1002, 2021.

[7] J. Ureña, A. Hernández, J. J. García, J.M. Villadangos, M.C. Pérez, D. Gualda, F. J. Álvarez, and T. Aguilera, "Acoustic Local Positioning With Encoded Emission Beacons," Proceedings of the IEEE, vol. 106, no. 6, pp. 1042-1062, 2018.

[8] C.Medina, J.C.Segura, A.De la Torre, “TELIAMADE: sistema de localización en interiores basado en ultrasonidos y RF”.[En línea]. Disponible en: https://www.researchgate.net/publication/259469742_TELIAMADE_Sistema_de_localizacion_en_interiores_basado_en_ultrasonido_y_RF. [Consultado: 01-jun-2021].

[9] D. Rodríguez-Navarro, J. L. Lázaro-Galilea, A. De-La-Llana-Calvo, A. Gardel-Vicente, I. Bravo-Muñoz, y F. Espinosa-Zapata, “Posicionamiento en Interiores de Alta Precisión Utilizando Tecnología Infrarroja”, Uah.es. [En línea]. Disponible en: https://ebuah.uah.es/dspace/bitstream/handle/10017/39788/Posicionamiento_Rodr%C3%ADguez_Navarro_SAAEI_2019_%282%29.pdf?sequence=3&isAllowed=y. [Consultado: 01-jun-2021].

[10] M. Groth, K. Nyka, y L. Kulas, “Calibration-free single-anchor indoor localization using an ESPAR antenna”, Sensors (Basel), vol. 21, núm. 10, p. 3431, 2021.

[11] F. S. Granja, “Tema 3.3: Sistemas de localización en interiores basados en radiofrecuencia”, Upm-csic.es. [En línea]. Disponible en: http://www.car.upm-csic.es/lopsi/static/publicaciones/docencia/Apuntes%20RF-LPS.pdf. [Consultado: 01-jun-2021].

[12] N. E. J. P. V. I. T. de I. de sistemas Sistemas Empotrados, “Sistema de posicionamiento móvil para interiores vía WIFI”, Uoc.edu. [En línea]. Disponible en: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/53021/7/pfi45js390pTFC0716mem%C3%B2ria.pdf. [Consultado: 01-jun-2021].

[13] T. H. Do, and M. Yoo, “Performance Analysis of Visible Light Communication Using CMOS Sensors,” Sensors, vol. 16(3), no. 309, pp. 1-23, 2016.

[14] N. Rajagopal, P. Lazik, A. Rowe, “Visual light landmarks for mobile devices,” Proceedings of the 13th International Symposium on Information Processing in Sensor Networks, pp. 249–260, 2014.

[15] C. Danakis, M. Afgani, G. Povey, I. Underwood, and H. Haas, “Using a CMOS camera sensor for visible light communication,” Proceedings of the Globecom Workshops (GC Wkshps), pp. 1244–1248, 2012.

[16] G. Simon, G. Zachár and G. Vakulya, "Lookup: Robust and Accurate Indoor Localization Using Visible Light Communication," in IEEE Transactions on Instrumentation and Measurement, vol. 66, no. 9, pp. 2337-2348, Sept. 2017.

[17] P. H. Pathak, X. Feng, P. Hu, and P. Mohapatra, "Visible Light Communication, Networking, and Sensing: A Survey, Potential and Challenges," IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 2047-2077, 2015.

[18] S. Cincotta, C. He, A. Neild, and J. Armstrong, "High angular resolution visible light positioning using a quadrant photodiode angular diversity aperture receiver (QADA)," Optics Express, vol. 26, no. 7, pp. 9230-9242, 2018.

[19] “The Design Warrior’s Guide to FPGAs”, Clive “Max” Maxfield, Elsevier (2004).

[20] “Introduction to FPGA and it’s programming tools”, Circuitdigest.com, 24-abr-2019. [En línea]. Disponible en: https://circuitdigest.com/tutorial/what-is-fpga-introduction-and-programming-tools. [Consultado: 01-jun-2021].

[21] V. 2., “(Zynq ™ Evaluation and Development) Hardware User’s Guide”, Element14.com. [En línea]. Disponible en: https://www.element14.com/community/servlet/JiveServlet/previewBody/90929-102-1-377430/ZedBoard_HW_UG_v2_2.pdf. [Consultado: 01-jun-2021].

[22] L. H. Crockett, R.A. Eliot, M. A. Enderwitz, R.W. Stewart, “The Zynq Book: Embedded Processing with the ARM Cortex -A9 on the Xilinx Zynq-700 All Programmable SoC”. Departamento of Electronic and Electrical Engineering, University of Strathclyde. Glasgow, Scotland, UK. 2014.

[23] T. Agarwal, “Application specific integrated circuit: Types, and applications”, Elprocus.com, 06-abr-2019. [En línea]. Disponible en: https://www.elprocus.com/application-specific-integrated-circuits/. [Consultado: 27-mayo-2021].

[24] C. del Capítulo, “Introducción a los DSP. Conceptos básicos 1”, Mheducation.es. [En línea]. Disponible en: https://www.mheducation.es/bcv/guide/capitulo/8448198344.pdf. [Consultado: 01-jun-2021].

[25] M. L. López Vallejo, J. L. Ayala Rodrigo, “FPGA: Nociones básicas e implementación”. Universidad Politécnica de Madri. Abril 2014. Disponible en: https://www.researchgate.net/profile/Marisa-Lopez-Vallejo-2/publication/266337852_FPGA_Nociones_basicas_e_implementacion/links/54b79b1c0cf2bd04be33ac8a/FPGA-Nociones-basicas-e-implementacion.pdf. [Consultado: 01-jun-2021].

[26] First Sensor Inc., Series 6 Data Sheet Quad Sum and Difference Amplifier, Part Description QP50-6-18u-SD2, Product Specification, 2012.

[27] Xilinx.com. [En línea]. Disponible en: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_1/ug906-vivado-design-analysis.pdf.

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR