UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA

Desarrollo de un Sistema para la caracterización de la transmisión OOK-NRZ a través de LEDs de iluminación

Autor: Adrián Luis Arándiga Martínez

Tutor: Javier Valls Coquillat y Vicente Torres Carot

Trabajo Fin de Máster presentado en el Departamento de Ingeniería Electrónica de la Universitat Politècnica de València para la obtención del Título de Máster Universitario en Ingeniería de Sistemas Electrónicos

Curso 2020-21

Valencia, julio de 2021

Resumen

En este trabajo se realiza el diseño e implementación en FPGA de un sistema que permite evaluar una transmisión a través de LEDs de iluminación mediante el uso de la modulación OOK-NRZ. El sistema permitirá operar con diferentes frecuencias de transmisión y capturar la señal con diferentes fases. Se realizará una calibración para seleccionar la fase óptima y, capturando la señal en esa fase, se medirá la tasa de error de bit de la transmisión. Se implementará un interfaz para controlar el sistema desde un ordenador mediante instrucciones en Matlab. Finalmente, se montará un prototipo y se realizarán medidas para caracterizar el sistema.

Resum

En aquest treball, es realitza el disseny i l'implementació en FPGA d'un sistema que permet avaluar una transmissió a través de LED's d'il·luminació per mitjà de l'ús de la modulació OOK-NRZ. El sistema permetrà operar amb diferents freqüències de transmissió i capturar el senyal amb diferents fases. Es realitzarà una calibració per a seleccionar la fase òptima i, capturant el senyal en eixa fase, es mesurarà la taxa d'error de bit de la transmissió. S'implementarà un interfície per a controlar el sistema des d'un ordinador per mitjà d'instruccions en Matlab. Finalment, es muntarà un prototip i es realitzaran mesures per a caracteritzar el sistema.

Abstract

The main goal of this work is the design and implementation on an FPGA device of a system that evaluates a transmission through illumination LEDs using the OOK-NRZ modulation. The system will be able to operate with different transmission frequencies and capture the signal with different phases. The optimal phase will be decided with a calibration stage. That phase will be used to capture the signal to measure the bit error rate of the transmission. An interface to control the system from a computer through Matlab instructions will be implemented as well. Finally, the prototype will be mounted and tested.

Índice

Índice de figuras

Índice de tablas

Listado de acrónimos

BER: Bit Error Rate. Tasa de error de bit.

FIFO: First In, First Out. Primero en entrar, primero en salir.

FPGA: Field-Programmable Gate Array. Matriz de puertas lógicas programables en campo.

HDL: Hardware Description Language. Lenguaje de descripción hardware.

HPS: Hard Processor System.

IM/DD: Intensity Modulation and Direct Detection. Modulación de intensidad y detección directa.

IP: Intellectual Property. Propiedad intellectual.

IP: Internet Protocol. Protocolo de internet.

LED: Light-Emitting Diode. Diodo emisor de luz.

LFSR: Linear Feedback Shift Register. Registro de desplazamiento con retroalimentación lineal.

OOK-NRZ: On-Off Keying Non-Return to Zero. Manipulación Encendido-Apagado sin retorno a cero.

PC: Personal Computer. Ordenado personal.

PLL: Phase-Locked Loop. Lazo de seguimiento de fase.

RAM: Random Access Memory. Memoria de acceso aleatorio.

RTL: Register Transfer Level. Transferencia a nivel de registro.

SNR: Signal-to-Noise Ratio. Relación señal ruido.

TCP: Transmission Control Protocol. Protocolo de control de transmisión.

TFM: Trabajo Fin de Master.

VLC: Visible Light Communication. Comunicación con luz visible.

Capítulo 1. Introducción

1.1 Introducción

En este proyecto, se pretende proporcionar una herramienta que permita testear comunicaciones con luz visible.

La comunicación con luz visible (VLC) utiliza LEDs como fuente de iluminación, y como transmisor de señal modulando los LEDs. Por lo que con VLC se combinan las funcionalidades de iluminación y comunicación [1].

La comunicación con luz visible ofrece ciertas ventajas sobre la comunicación por radio frecuencia. No tienen interferencia electromagnética al tratarse de comunicaciones con luz. Tienen bajo coste ya que reutilizan la infraestructura de iluminación, añadiéndole ciertos módulos. Son energéticamente eficientes ya que utilizan los LEDs como fuente de transmisión. Y proporcionan seguridad en la transmisión ya que la luz no penetra objetos opacos [2].

No obstante, también pueden mencionarse diversas desventajas. VLC únicamente funciona cuando la iluminación está activada. Sin una fuente de iluminación directa, la transferencia de datos se detiene. La luz natural puede afectar negativamente a la comunicación, pero reducir el aprovechamiento de la luz natural entra en conflicto con el aspecto del ahorro energético de la VLC. La aplicación en dispositivos portátiles o en exteriores todavía está en vías de investigación. Y el ancho de banda está limitado por los LEDs comerciales.

Algunas de las desventajas de la comunicación con luz visible podrían mitigarse combinando VLC con Wi-Fi. Las aplicaciones de la VLC donde se aprovechan sus puntos fuertes, son redes en el hogar, oficinas, clases y hoteles entre otras, donde el receptor podría estar en un punto fijo bajo la fuente de iluminación. En ambientes industriales pueden ser particularmente convenientes debido a su inmunidad electromagnética. De forma parecida, es posible utilizar VLC en zonas sensibles a interferencia electromagnética, como hospitales y cabinas de avión [3].

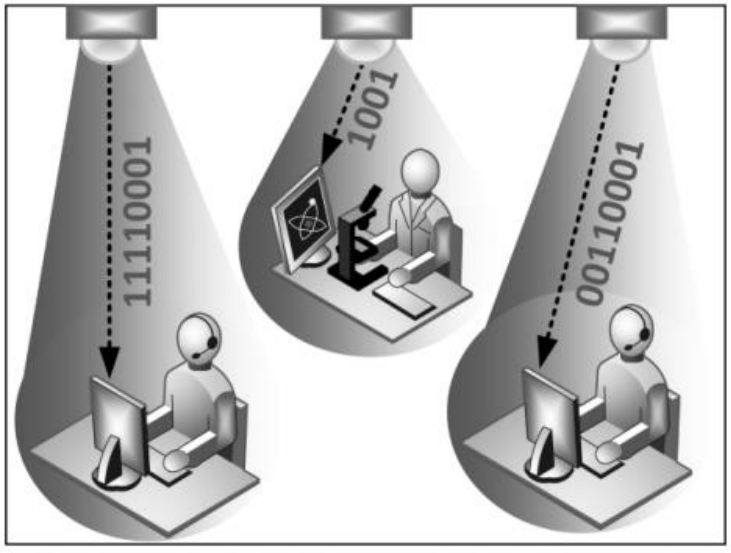


Figura 1.1 VLC en un entorno de oficina. Fuente [3].

1.2 Modulación OOK-NRZ en la comunicación con luz visible

El sistema diseñado en este proyecto se utiliza para evaluar la transmisión a través de LEDs de iluminación mediante el uso de la modulación on-off keying non return to zero (OOK-NRZ). Consiste en una modulación de amplitud, donde se conmuta entre dos niveles de tensión la fuente de iluminación para transmitir un uno o un cero, y después de enviar un bit de información se envía inmediatamente el siguiente.

La transmisión y recepción de información se realiza por modulación de intensidad y detección directa (IM/DD). Para ello, un modulador transforma el flujo de bits a transmitir en una señal analógica. Al modulador le sigue un driver que amplifica la señal y alimenta la fuente de iluminación. Esta emite pulsos de luz que se transmiten a través del aire y son capturados por un fotodetector, que vuelve a transformar los pulsos de luz en señal eléctrica. Esta señal pasa por una etapa de amplificación y filtrado que se conecta a un demodulador para recuperar la señal transmitida. La señal por encima de una tensión determinada se considera un uno lógico, y por debajo un cero. En la Figura 1.2 se muestra la estructura de un sistema de VLC basado en modulación de intensidad y detección directa [4].

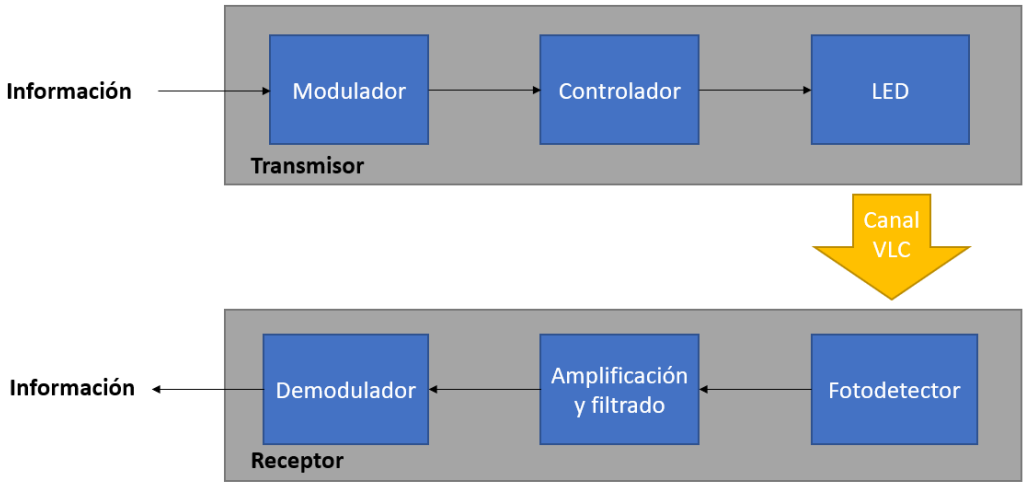


Figura 1.2 Estructura de sistema VLC basado en (IM/DD).

El medio de transmisión y el procesado de la señal introducen un retardo entre la señal transmitida y la recibida. Además, los pulsos recibidos se verán deformados en función de la frecuencia de transmisión. La recepción de los pulsos se puede observar en la Figura 1.3. Para asegurar una lectura de la señal correcta, cada pulso se captura en distintas fases. En una etapa de calibración se encuentra cual de esas fases es la que captura mejor la señal y se realiza la recepción de la señal en esa fase.

Figura 1.3 Señal recibida a 2.5 Mb/s.

1.3 Objetivos

El objetivo de este TFM es el de desarrollar con un dispositivo FPGA un sistema que permita transmitir una trama de datos modulados en OOK-NRZ, a través de un LED de iluminación, y que, al recibirlos mediante un fotodiodo, recupere la fase de transmisión, sincronice el inicio de la trama y realice una medida del bit error rate (BER). Se monta un prototipo y se realizan medidas para caracterizar la transmisión.

La lista de objetivos a cumplir es la siguiente:

- Desarrollo de un modelo del sistema en Matlab. Incluye el modelo del canal, la etapa de calibración para obtener la fase óptima y la etapa de medida del BER.
- Implementación del sistema en SystemVerilog. Este objetivo incluye el diseño de los módulos, su codificación en lenguaje de descripción hardware (HDL) y verificación.
- Desarrollar la etapa de comunicación del sistema con el PC.
- Montaje del prototipo y realización de pruebas.

1.4 Herramientas y metodología

1.4.1 Placa de desarrollo

El prototipo se implementa en la placa de desarrollo DE10-Standard de Terasic [5] que integra una FPGA Cyclone V SX SoC – 5CSXFC6D6F3F31C6N. En la Figura 1.4 se muestran los periféricos que incluye la placa.

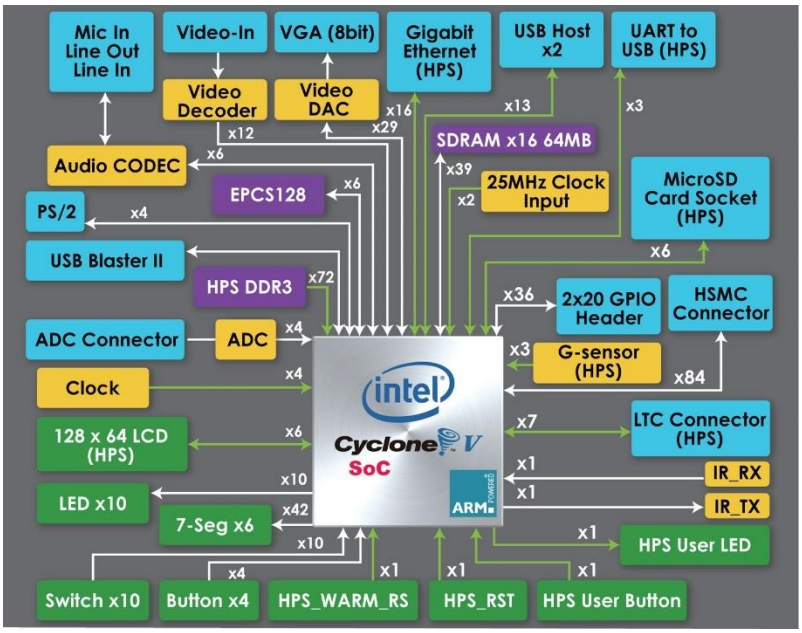


Figura 1.4 Periféricos DE10-Standard. Fuente [5].

Los que se utilizan en el prototipo son:

- Conexión Ethernet para la comunicación con el PC.
- LEDs como indicadores.
- Interruptores como señales de control.
- Pulsadores como señales de reset.
- GPIO Para la transmisión y recepción de señal.

1.4.2 Matlab

Es una software de programación y cálculo numérico. Matlab permite realizar manipulación de matrices, representación gráfica de datos y la implementación de algoritmos entre otras funciones.

Se ha utilizado para modelar el sistema y estudiar el comportamiento del mismo para diferentes parámetros y niveles de ruido. Además, a partir de los modelos, se han generado los ficheros utilizados en la verificación de los módulos hardware y se ha creado el paquete con los parámetros de configuración del hardware. Finalmente, se ha utilizado como interfaz de usuario entre el PC y el sistema para el control y lectura de registros.

1.4.3 Quartus Prime

Software de diseño de dispositivos de lógica programable producido por Intel. Permite realizar el análisis y síntesis de diseños HDL. Se ha utilizado para la implementación del hardware en SystemVerilog.

1.4.4 ModelSim

Software de simulación de HDL que permite realizar simulaciones a nivel de transferencia de registros (RTL) y a nivel de puertas lógicas. Se ha empleado para la verificación de todos los módulos hardware diseñados, tanto individualmente, como en conjunto al integrarlos en bloques.

1.4.5 Platform Designer

Herramienta de integración de sistemas que permite generar automáticamente lógica para conectar funciones de propiedad intelectual (IP) con subsistemas. Se ha utilizado para integrar el sistema en un componente que permite la comunicación de manera sencilla con el microprocesador de la placa de desarrollo.

1.4.6 SignalTap Logic Analyzer

Es una herramienta de depuración a nivel de sistema que permite monitorizar distintas señales y registros internos del diseño sintetizado en la FPGA. Se sintetiza junto al sistema a depurar y consume recursos lógicos de la FPGA. Se ha utilizado para la depuración del sistema completo implementado en la FPGA, visualizando la evolución de las señales y el valor de los registros.

1.4.7 Visual Studio

Es un entorno de desarrollo integrado compatible con múltiples lenguajes de programación. Se ha utilizado para establecer la comunicación entre la placa de desarrollo y el PC.

1.4.8 Metodología de trabajo

Primero se modela utilizando Matlab la calibración y el cálculo del BER. Este modelo permite que se entienda el funcionamiento del sistema y la realización de pruebas para ver como la modificación de diferentes parámetros de configuración afectan al mismo. Con la información obtenida, se decide el criterio para considerar una fase válida y cómo elegir la fase óptima de entre las válidas.

Para el diseño del hardware, se separan las funciones que debe cumplir el sistema en distintos bloques que se coordinan con un control general. Cada uno de estos bloques, se separa en los módulos necesarios para que el conjunto de estos realice una función concreta. Estos bloques cuentan con su propio control. En la Figura 1.5 se muestra la compartimentalización del sistema. Se diseña siguiendo la filosofía de usar máquinas de estado de alto nivel descritas en el capítulo 5 del libro de diseño digital de Frank Vahid [6]. Con el comportamiento del módulo definido, se diseña la ruta de datos necesaria para llevar a cabo las operaciones necesarias, y se conecta esta ruta de datos al control.

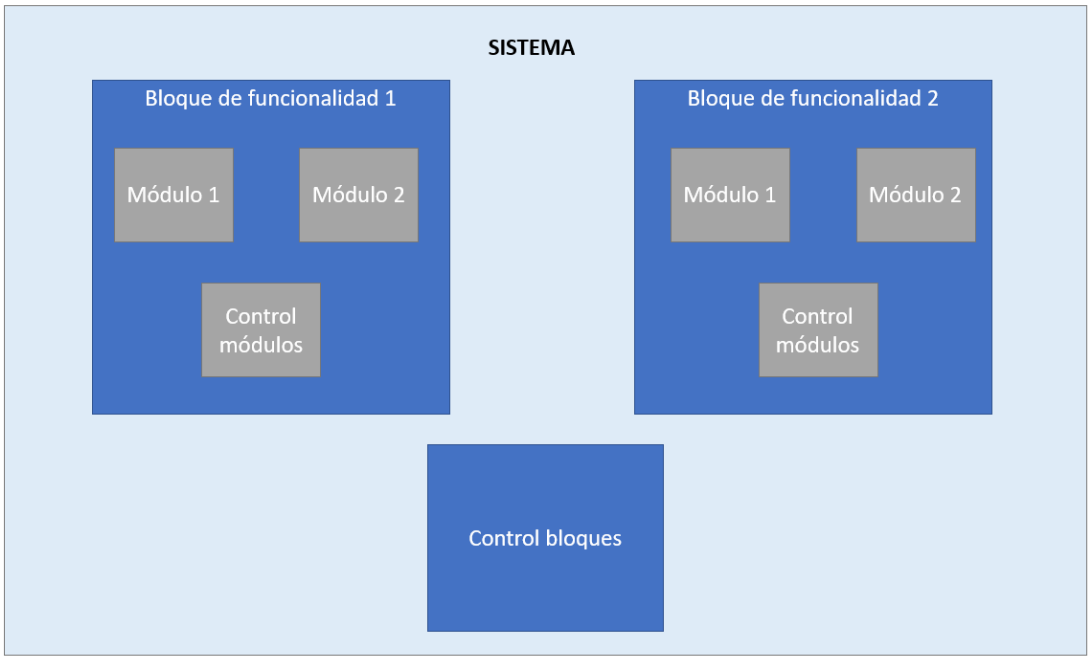


Figura 1.5 Estructura modular del sistema.

La mayoría de los módulos que componen el sistema se modelan en Matlab primero. De estos modelos se extraen ficheros utilizados en la verificación del módulo. A partir de este modelo se implementa el hardware en SystemVerilog. Los módulos que no siguen este flujo son los que utilizan una IP del catálogo de IP de Quartus Prime.

Una vez completado el hardware de un módulo, se realiza la verificación del sistema en ModelSim. Para la verificación se comparan los resultados del hardware con los obtenidos en Matlab y se observa que la evolución de las señales en el visor de formas de onda es coherente. En los módulos donde no haya datos que contrastar, se realiza la verificación analizando la evolución de las señales en la simulación. Para módulos donde la temporización pueda ser crítica, se realiza una simulación a nivel de puerta.

Una vez se han implementado y testeado todos los módulos que componen un bloque, se realiza la integración de los módulos en el bloque y se testea el bloque completo. Para estos bloques, se realizan pruebas más exhaustivas, modificando todos los parámetros de configuración del bloque para garantizar un funcionamiento robusto. Como en la verificación de los módulos, se contrastan los resultados del bloque hardware con el modelo en Matlab y se analiza la evolución de las señales.

Una vez implementados y testeados todos los bloques de funcionalidad, se integran en el sistema. Se verifica comparando los resultados del sistema con los del modelo Matlab para distintas configuraciones.

Con el testeo en ModelSim finalizado, se programa el prototipo en la placa de desarrollo y se utiliza el SignalTap para depurar el sistema y comprobar que la evolución de las señales y los datos almacenados en los registros son los esperados.

Finalmente se integra el sistema al equipo del laboratorio que realiza la transmisión de señales a través de un LED de iluminación y las recibe con un fotodetector. Se realizan pruebas de calibración y medida del BER a distintas frecuencias para comprobar el funcionamiento del sistema en una transmisión real.

1.5 Estructura de la memoria

La memoria se estructura como se explica a continuación:

- En el punto 2 se explica el sistema y su implementación hardware. Primero se muestra el sistema completo. A continuación, se subdivide en los distintos bloques que realizan las funciones y se detalla su implementación. Estos se descomponen en diversos módulos que se utilizan para realizar la función del bloque y también son explicados.
- En el punto 3 se detalla cómo se ha realizado la interfaz que permite al usuario comunicarse con el sistema desde un PC a través de Matlab.
- En el punto 4 se muestran los resultados de test del prototipo programado en la placa de desarrollo. Primero se muestran los resultados obtenidos realizando pruebas con el SignalTap. Luego se explica el sistema de transmisión con LED del laboratorio, y se muestran los resultados de las pruebas del sistema dentro del entorno del laboratorio.
- En el punto 5 se exponen las conclusiones.
- En el punto 6 se describen ampliaciones y posibles mejoras del sistema.

Capítulo 2. Diseño del sistema

2.1 Funcionamiento del sistema

El sistema realiza dos funciones principales, la calibración y el cálculo del BER. A través de un interfaz se inicia la etapa de calibración, cuando finaliza esta etapa, se pasa a la etapa del cálculo del BER. Cuando acaba con el cálculo, se almacenan los resultados en registros y finaliza el proceso. El diagrama de flujo del sistema, teniendo en cuenta únicamente estas dos etapas, se muestra en la Figura 2.1. Además de esto, el sistema genera la señal a transmitir.

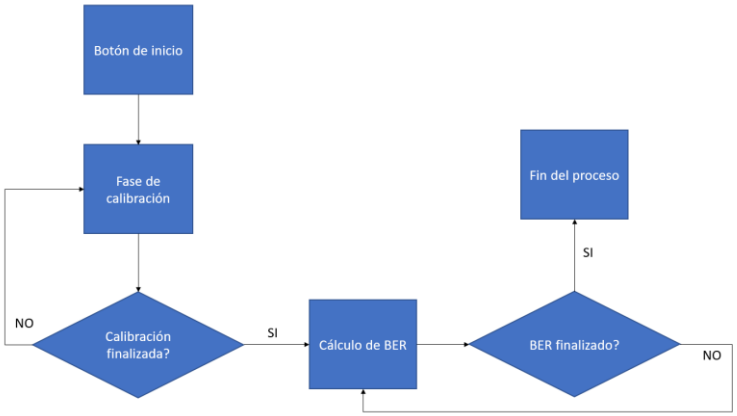


Figura 2.1 Diagrama de flujo del sistema.

2.1.1 Etapa de calibración

En el modo de calibración, se configura el sistema para que se capture la señal en la parte óptima del pulso con el bit recibido para minimizar las pérdidas de datos por el deterioro de la señal en la transmisión.

Para conseguir esto, el sistema emite una secuencia de calibración. Cada bit de esta secuencia se captura en varias fases del reloj de recepción de señal. En todas las fases, se realiza una operación de correlación de la secuencia recibida con la secuencia de calibración. La secuencia de calibración se envía un número elevado de veces para cada una de las fases y se realiza el promediado de los máximos de correlación de cada fase. Finalmente se realiza la decisión de fase óptima, seleccionando la fase intermedia entre todas las que superan cierto nivel de correlación. En la Figura 2.2 se muestra el diagrama de bloques de la fase de calibración.

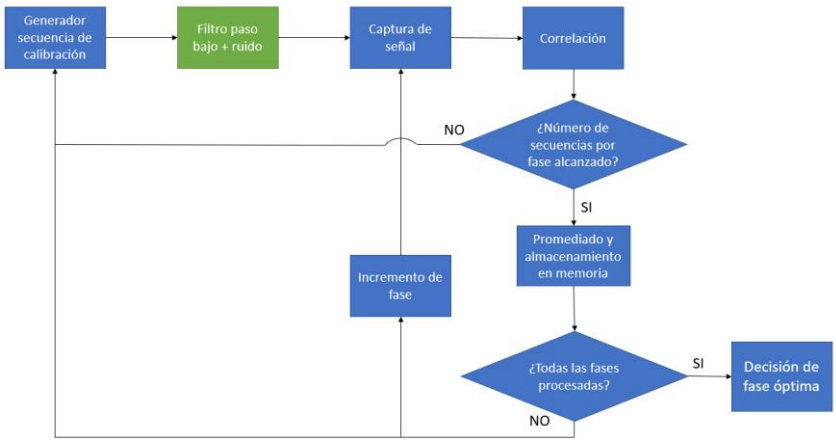


Figura 2.2 Diagrama de bloques fase de calibración.

El bloque de calibración se implementa primero en Matlab para tener un modelo con el que comprobar y estudiar el comportamiento de esta etapa y que, posteriormente, se utiliza para la verificación de la implementación hardware.

En esta etapa primero se generan múltiples secuencias de calibración, utilizando un registro de desplazamiento con retroalimentación lineal, también conocido como linear feedback shift register (LFSR). En Matlab la secuencia se genera utilizando el objeto PNSequence de la caja de herramientas de comunicaciones. [7]. Para modelar el efecto del canal de transmisión, esta secuencia se filtra con un filtro paso bajo y se le introduce un ruido, el resultado de este proceso puede observarse en Figura 2.3. Se captura la secuencia recibida en distintas fases del reloj de recepción y se realiza la operación de correlación de cada fase de la señal filtrada con la secuencia de calibración. Esta operación se realiza múltiples veces para poder calcular el valor medio de correlación en cada fase. Con estos valores medios, se realiza la decisión de fase óptima.

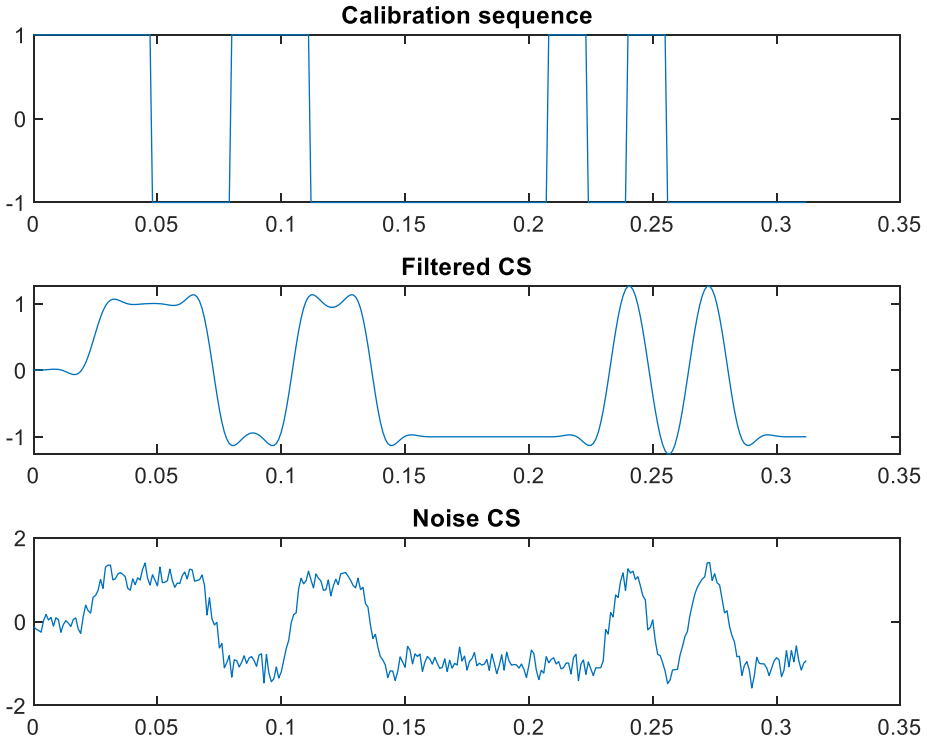


Figura 2.3 Señal de calibración. Arriba directa del LFSR. Medio Filtrada. Abajo añadiendo el ruido.

En la gráfica (a) de la Figura 2.4, se representa la media de la correlación cruzada entre las secuencias de calibración recibidas capturadas en ocho fases y la secuencia de calibración generada sin pasar por el medio de transmisión. La secuencia de calibración en esta gráfica es de quince bits. En el quinceavo desplazamiento de la operación de correlación, se compara la secuencia recibida completa con la generada y se puede observar cómo se alcanza el máximo de correlación.

La gráfica (b) de la Figura 2.4, recoge el valor medio de los máximos de correlación obtenidos en cada fase en un número elevado de secuencias de calibración. Con esta gráfica es posible determinar con qué fases del reloj de recepción se captura mejor la señal.

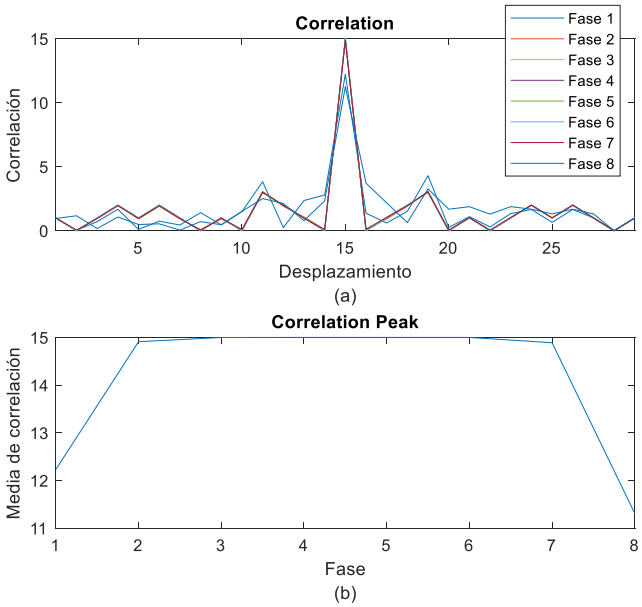


Figura 2.4 Curvas de correlación obtenidas en Matlab para 8 fases y secuencia de 15 bits (a) Correlación en cada fase. (b) Media de correlación por fase.

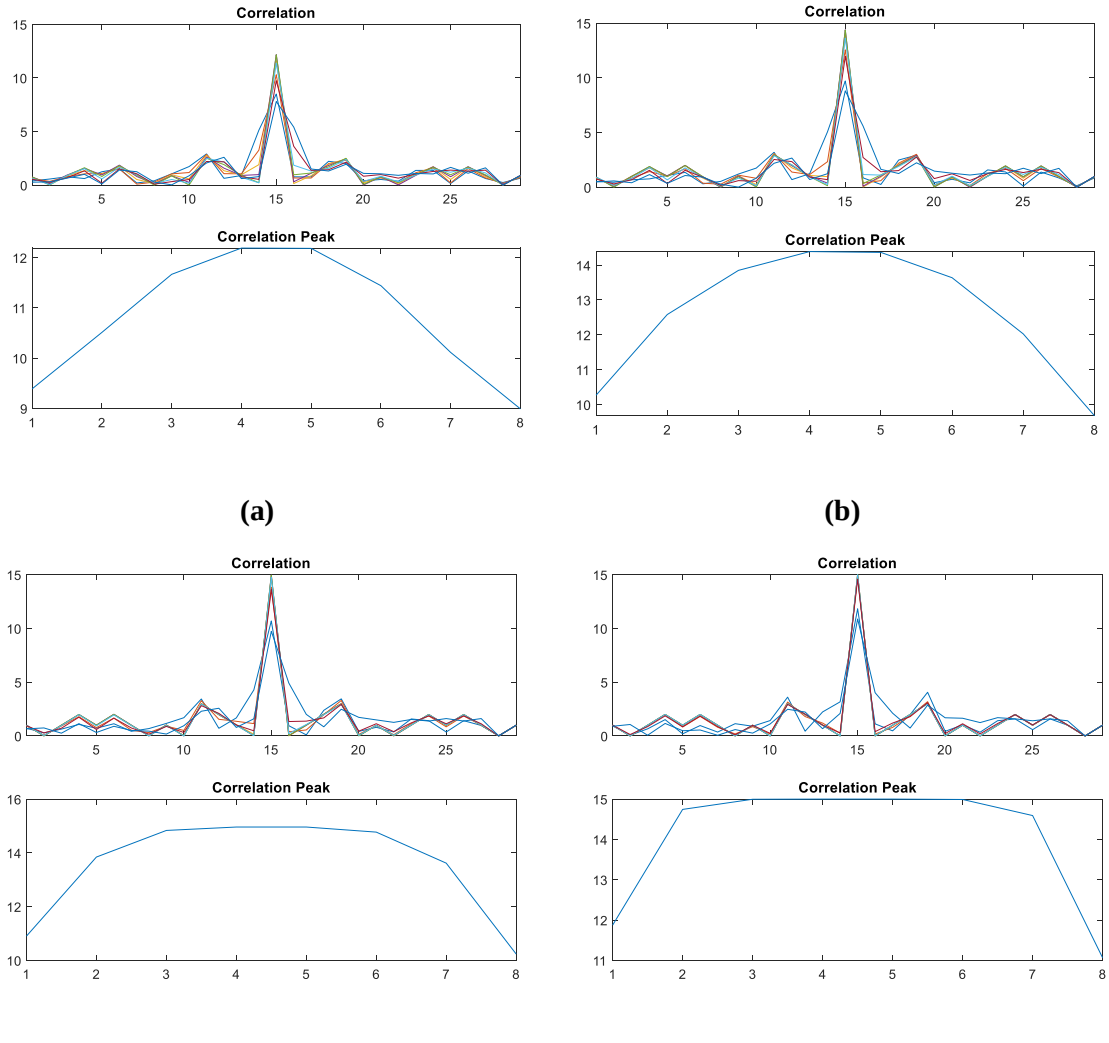


Figura 2.5 Correlación en función del ruido. (a) SNR_dB=4. (b) SNR_dB=8. (c) SNR_dB=12. (d) SNR_dB=16.

Utilizando el modelo de simulación de esta etapa, se estudia como el ruido afecta a estas curvas de correlación. En la Figura 2.5 se representa el resultado de la correlación para diferentes relaciones de señal y ruido (SNR). Puede observarse como para malas relaciones señal ruido, los máximos de correlación obtenidos están por debajo del máximo de correlación posible, y pocas fases alcanzan niveles altos de correlación. A medida que va mejorando la relación señal ruido, el máximo valor de correlación obtenido se va aproximando al máximo valor de correlación posible, hasta finalmente alcanzarlo. A su vez el número de fases con buena correlación va aumentando.

Como se puede ver, las peores fases se encuentran en los extremos y las mejores en el centro, por eso se selecciona la fase central de entre las que superan cierto nivel de correlación como fase óptima sobre la que capturar la señal recibida. De esta forma se sabe que se está capturando en una de las fases con mejor correlación, y se tiene cierto margen de maniobra.

Se han realizado distintas pruebas para comprobar como diferentes parámetros afectan a la calibración. En la Figura 2.6 se muestra como un filtro más estrecho reduce el número de fases válidas.

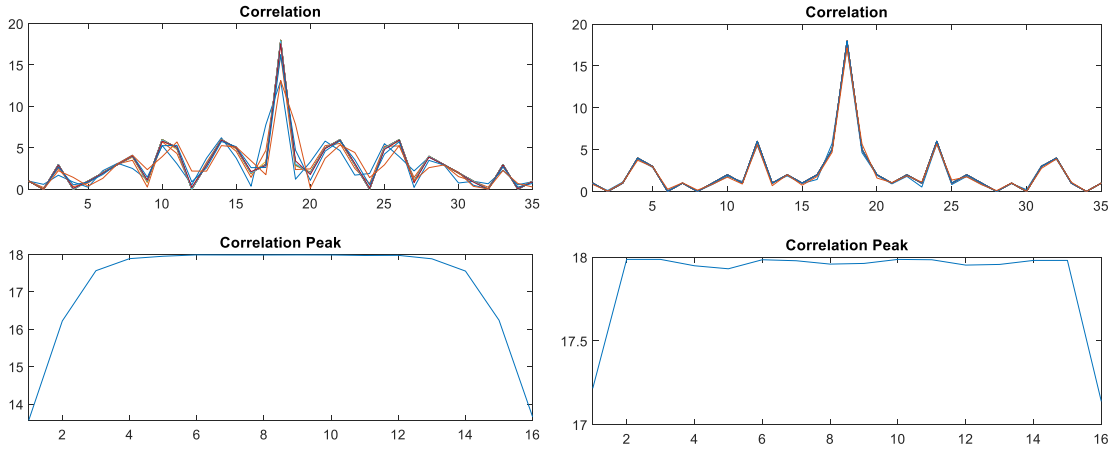


Figura 2.6 Curvas de correlación en función del filtro. Derecha wn = 1/2. Izquierda wn = 1/8.

Se estudia también como el número de fases con las que se captura la señal afecta al sistema, y se concluye que, a mayor número de fases, más amplio es el intervalo de fases válidas, como se observa en la Figura 2.7.

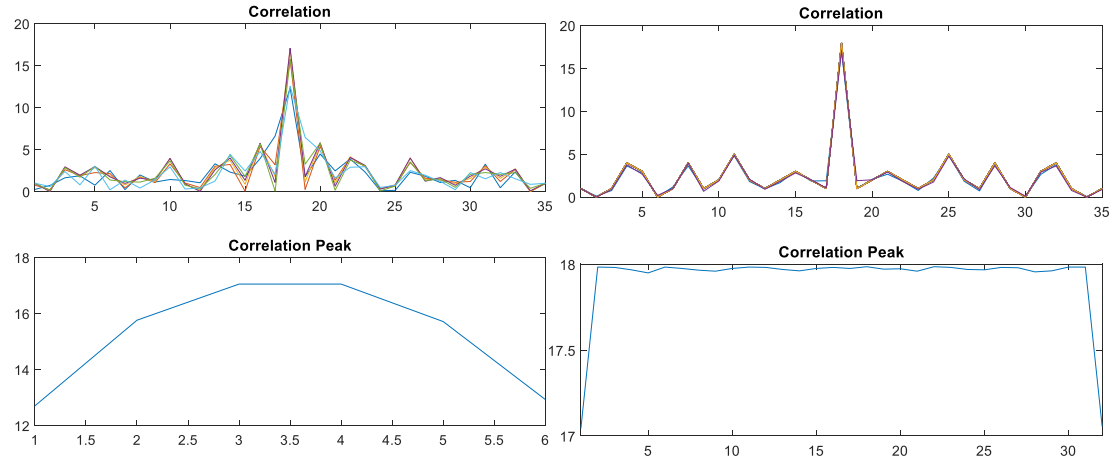


Figura 2.7 Correlación para 6 fases (Izqda.) y 32 (Dcha.) coeff 50, wn 1/2, snr 10.

El retardo introducido por el canal hace que la curva de correlación obtenida se desplace y se vea como se muestra en la Figura 2.8. Esto complica la decisión de la fase óptima ya que no es posible conocer antes de la transmisión en qué momento comienza a capturarse la señal ni qué forma tendrá la curva de correlación. Para solucionarlo, se almacena la curva de correlación de las secuencias recibidas en una memoria. A continuación, se ordena la curva comenzando por la fase de menor correlación y con esto ya es posible diseñar un módulo que seleccione la fase central de forma sencilla.

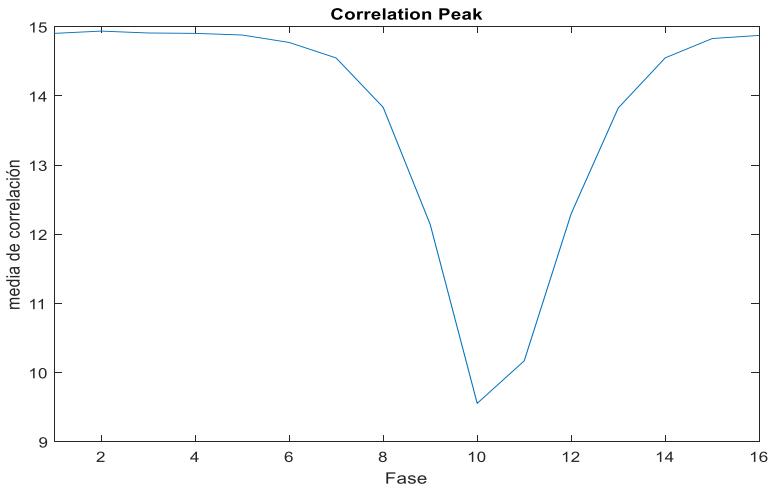


Figura 2.8 Curva correlación con un retardo de 10 muestras.

2.1.2 Etapa de cálculo del BER

En el modo de cálculo del BER se compara la señal recibida con la misma señal sin pérdidas por transmisión y se contabiliza el número de errores y el número de bits recibidos para obtener el BER

Para ello, se genera una secuencia compuesta de una cabecera para indicar el inicio de la transmisión y la señal a transmitir en sí, que se trata de una secuencia pseudoaleatoria. Se captura la señal en la fase óptima obtenida previamente en la etapa de calibración. La seña recibida pasa por un correlador configurado con la secuencia de la cabecera. Con esto se consigue detectar el inicio de la trama cuando el valor a la salida del correlador supere el umbral de correlación calculado en la fase de calibración. Así que el cálculo del BER comienza a partir de la recepción de la cabecera. A partir de este punto se realiza el cálculo del BER comparando la secuencia aleatoria recibida tras el preámbulo con la misma secuencia generada internamente por el sistema. En la Figura 2.9 se muestra el diagrama de bloques de la etapa del cálculo del BER.

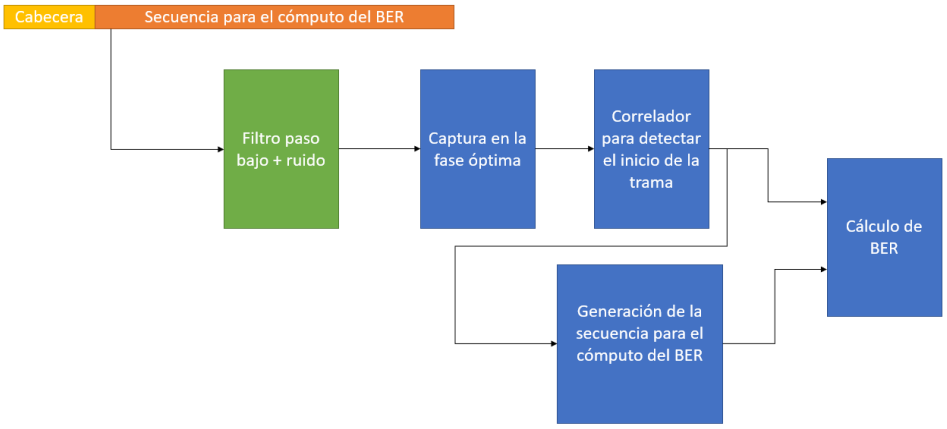
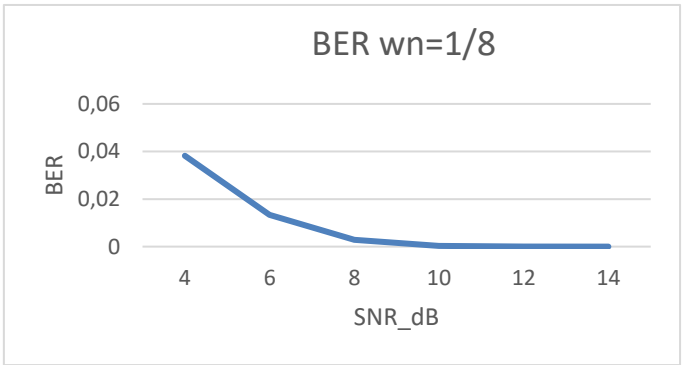


Figura 2.9 Diagrama de bloques cálculo del BER.

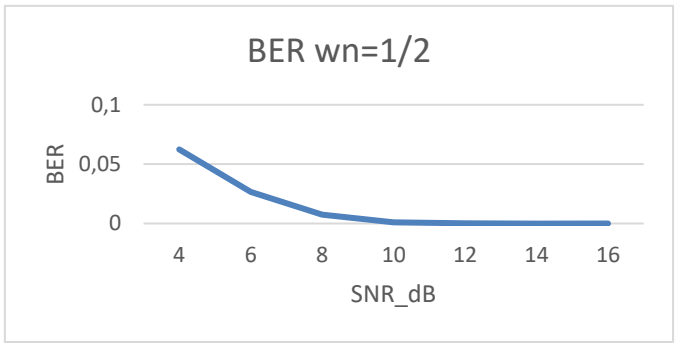
De la misma forma que se realizó en la etapa de calibración, el bloque del cálculo del BER se implementa primero en Matlab para comprender el funcionamiento de esta etapa y para conseguir un modelo sobre el que comparar los resultados en la verificación del hardware.

El primer paso de esta etapa es generar la secuencia que se transmite para realizar el cálculo del BER. Esta secuencia se genera con un LFSR de la misma forma que en la etapa de calibración. Se utiliza como cabecera la señal utilizada como secuencia de calibración y se le concatena la generada por el LFSR. Se filtra y se añade ruido a esta señal y se captura con la fase óptima. Se compara con la secuencia generada directamente por el LFSR para realizar el cálculo del BER

Se realizan pruebas modificando la configuración de parámetros para comprobar cómo afectan al BER. En la Figura 2.10 se muestra la curva del BER en función de la SNR de dos sistemas cuyos canales están modelados uno con un filtro de frecuencia de corte wn = ½ (caso b) y otro cuatro veces menor (caso a). Se observa que con el filtro más estrecho se reduce el BER.



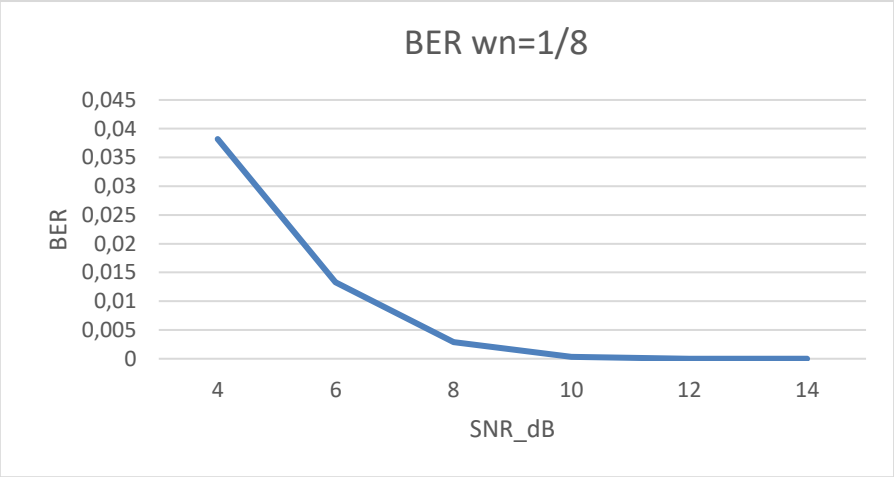
(a)



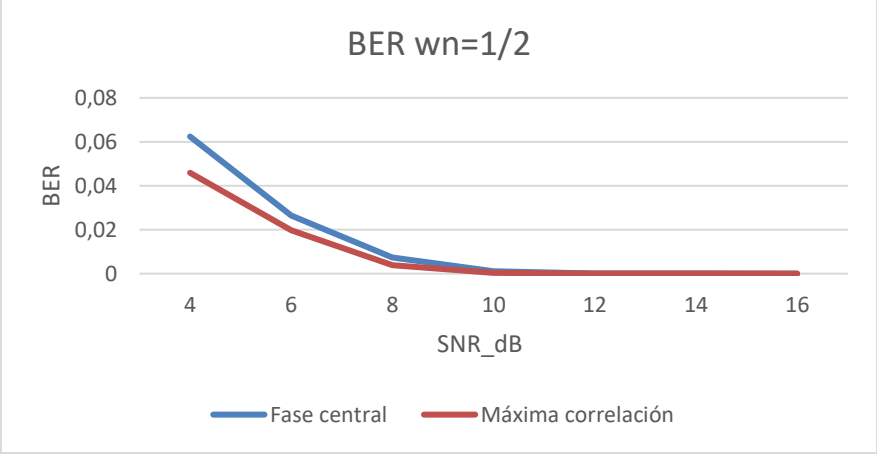
(b)

Figura 2.10 BER en función del ruido. (a) wn=1/8. (b) wn=1/2.

Se comprueba también la diferencia en el BER si se elige como fase óptima la fase central de entre las que superen un umbral o la fase con mayor correlación. Como se muestra en la Figura 2.11, para filtros más restrictivos la fase de mayor correlación coincide con la central. Con filtros menos restrictivos, se consigue mejor BER si se selecciona la fase de mayor correlación. El problema es que, si se selecciona la fase de mayor correlación, no se sabe exactamente dónde va a estar dentro de la curva de correlación, con lo que puede darse el caso en que esta fase esté adyacente a una fase donde la captura de señal sea mala. Es preferible capturar sabiendo que las fases que rodean a la seleccionada son también válidas.



(a)



(b)

Figura 2.11 Cálculo del BER comparando la fase central con la de máxima correlación para distintos filtros. (a) wn=1/8. (b) wn = 1/2.

2.2 Especificaciones técnicas

Como se ha comentado en el punto anterior, en la calibración se captura la señal con el mismo reloj de transmisión, pero desfasado en un número de fases diferentes y se selecciona la fase óptima para la captura de información. Para ello se promedia la recepción de varias secuencias de calibración. Teniendo en cuenta la funciones a realizar en esta parte, la configuración del sistema debe ser parametrizable tal y como se indica a continuación:

- Configuración de la secuencia de calibración: Se puede modificar el número de bits de esta secuencia y con el mismo número de bits, se pueden generar secuencias distintas.
- Número de fases para la captura de señal.
- Número de envíos de la señal de calibración por cada fase: Se envía la secuencia de calibración múltiples veces para realizar un promedio de todas las secuencias y minimizar la varianza introducida por el ruido.
- Cuenta de muestras por secuencia de calibración: Para compensar retardos generados por el canal de transmisión, se introduce una secuencia de ceros que se incluyen tras la secuencia de calibración, con esto se asegura que toda la secuencia de calibración es capturada.

En cuanto al cálculo del BER, como se ha comentado anteriormente, compara la secuencia recibida con una idéntica generada dentro del sistema. Por lo que los parámetros de configuración necesarios son los siguientes:

- Configuración de la secuencia transmitida: Se puede modificar el número de bits de esta secuencia y con el mismo número de bits, se pueden generar secuencias distintas.
- Número de errores a alcanzar: El cálculo del BER, se realiza hasta alcanzar el número de errores determinado en este parámetro.
- Configuración del contador: El sistema se diseña teniendo en mente que el número de muestras a contabilizar para realizar el cálculo del BER es elevado, por ello y para mejorar la frecuencia máxima del sistema, el contador se implementa con varios contadores en cascada. El número de contadores y el tamaño de los mismos es parametrizable.

Para obtener distintas frecuencias, un PLL genera distintos relojes. En Matlab se calculan los parámetros para realizar el incremento de fase en cada uno de los relojes.

El sistema se conecta al PC a través de una red TCP/IP y el usuario interactúa con el prototipo a través de comandos en Matlab.

2.3 Implementación hardware

El hardware puede descomponerse en siete módulos que se encargan de realizar las acciones necesarias para conseguir la calibración de la señal recibida y el cálculo del BER:

- DATA_FIFO: Se utiliza para capturar la señal de entrada bajo un reloj desfasado, y enviarla al resto de bloques bajo un reloj a la misma frecuencia, pero siempre en la misma fase. Este bloque funciona con los dos relojes generados por el PLL. Con esto se consigue capturar la señal en distintas fases, mientras que el resto de módulos pueden funcionar bajo un mismo reloj.
- CALIBRATION_BLOCK: Realiza la fase de calibración, funciona bajo el reloj estático generado por el PLL.
- BER_BLOC: Realiza la fase del cálculo del BER, funciona bajo el reloj estático generado por el PLL.
- PLL: Genera los relojes utilizados por el sistema, funciona bajo un reloj de referencia de 50MHz.
- INCREMENT_PHASE: Gestiona las señales para realizar el incremento de fase del reloj dinámico del pll, funciona bajo el mismo reloj de referencia de 50MHz del PLL.
- TRANSMISOR: Genera la señal a transmitir a través del LED, funciona bajo el reloj estático generado por el PLL.
- MAIN_CONTROL: Gestiona el control del sistema, funciona bajo el reloj estático generado por el PLL.

En la Figura 2.12 puede observarse el esquema de conexiones entre los distintos bloques del sistema.

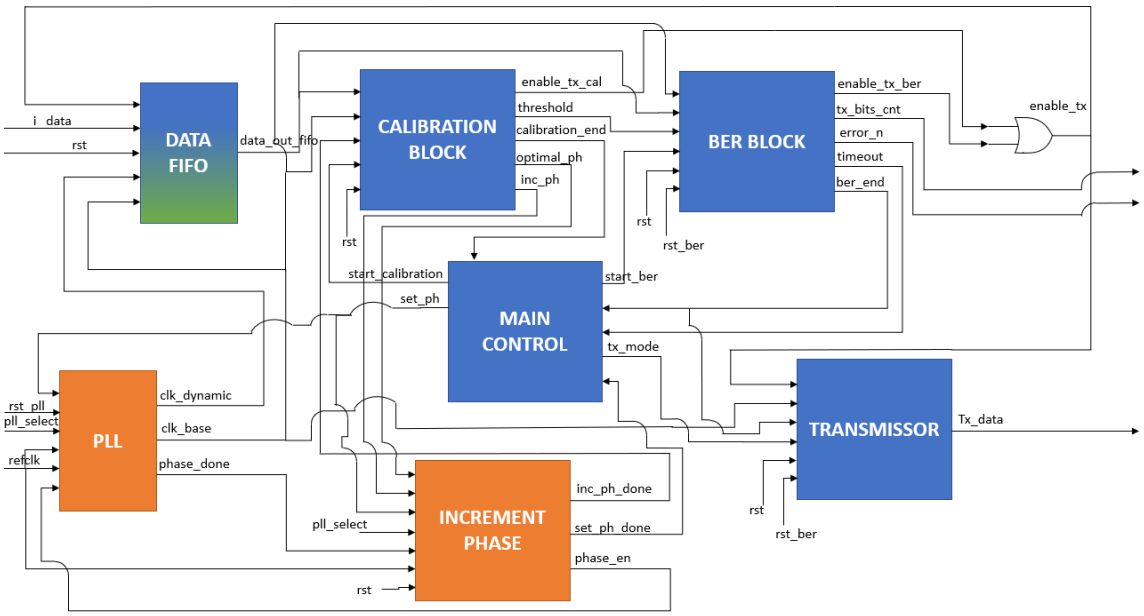


Figura 2.12 Conexiones entre los bloques que componen el sistema. Naranja: reloj de referencia. Azul: Reloj estático. Verde: Reloj dinámico.

Como interfaz de usuario, se utilizan tres botones para los resets del sistema, del pll y de la fase del cálculo del BER. Un switch para comenzar con el proceso de calibración, un switch o un botón para comenzar con el cálculo del BER y un switch para finalizar el proceso completo.

El estado del sistema se indica con LEDs, uno indica que el sistema está en el estado inicial y preparado para comenzar. Otro indica que se está realizando la calibración. Otro que se está esperando al input para pasar a la etapa del cálculo del BER. Mientras está realizando el BER, se ilumina un LED indicándolo. Finalmente se enciende un LED cuando el sistema ha finalizado o cuando ha habido un timeout.

Se configuran como registros de salida los resultados obtenidos de fase óptima, número de errores y número de muestras recibidas durante el cálculo del BER. En la Figura 2.13 se muestra la interfaz del sistema.

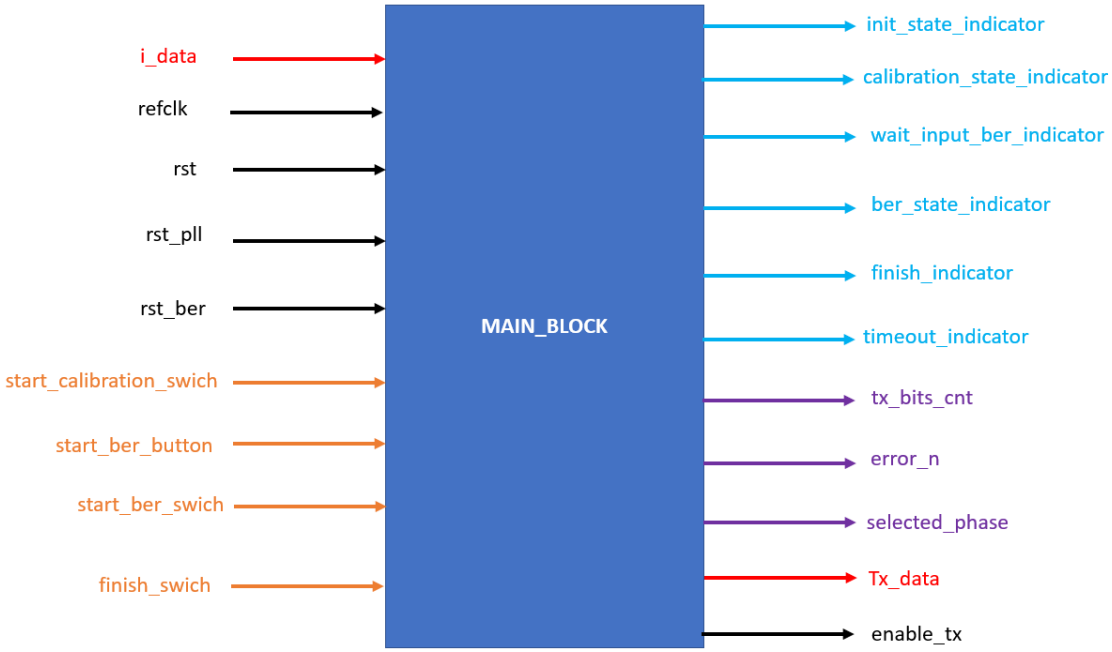


Figura 2.13 Interfaz del sistema.

Para la configuración del sistema, se genera un paquete desde Matlab con todos los parámetros que se incluye directamente en el proyecto de Quartus Prime.

2.3.1 Calibración

El bloque de calibración se encarga de seleccionar la fase óptima de entre todas las fases capturadas. Para ello se promedia la correlación de varias secuencias de calibración en una fase y se almacena el resultado obtenido en una memoria. A continuación, se activa una señal para solicitar el incremento de fase. Cuando se recibe la señal indicando que se ha completado el cambio de fase, se repite el proceso para el resto de las fases. Una vez se han procesado todas las fases, se calcula el umbral en función del promedio máximo y mínimo de correlación. Este umbral es utilizado para considerar una fase como válida. Finalmente se selecciona como fase óptima la fase intermedia de entre las que superan el umbral. Cada una de estas funcionalidades se implementa con distintos módulos.

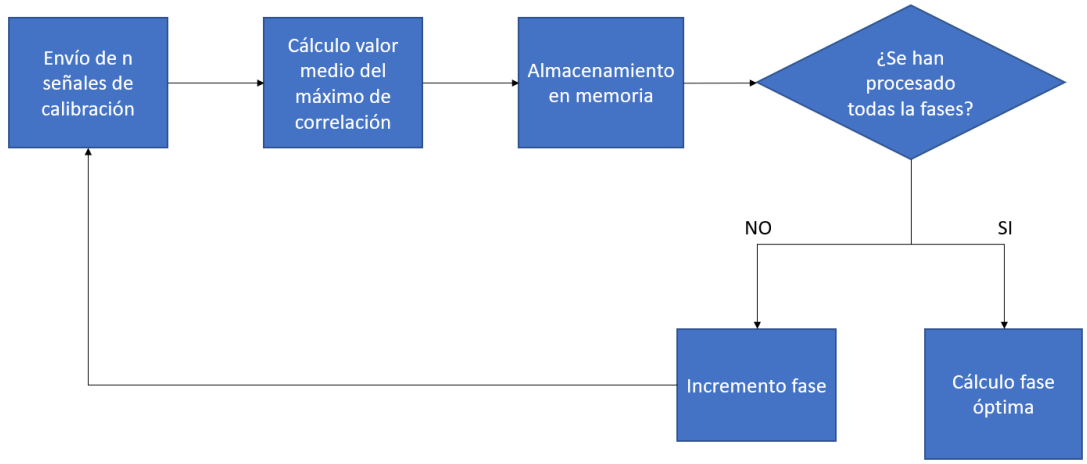


Figura 2.14 Diagrama de flujo proceso de calibración.

El esquema con las conexiones entre los módulos que componen el bloque CALIBRATION_BLOCK puede observarse en la Figura 2.15.

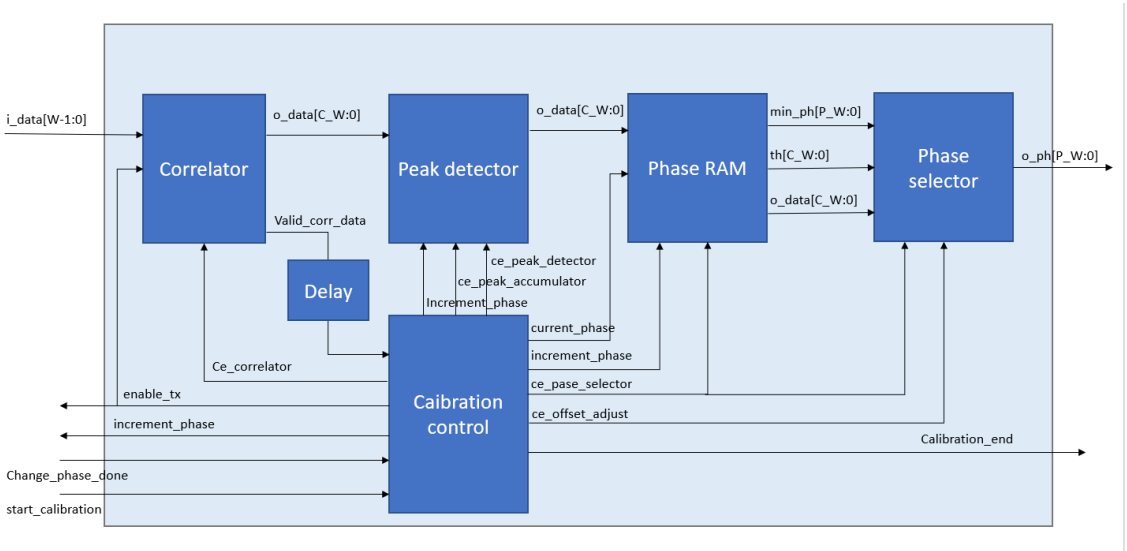


Figura 2.15 Esquema conexiones bloque CALIBRATION_BLOCK.

El correlador realiza la correlación de la señal recibida con la secuencia de calibración y envía el resultado a Peak_detector donde acumula los máximos de cada secuencia. Una vez se reciben todas las secuencias de una de las fases, se dividen los máximos acumulados en Peak_detector por el número de veces que se ha recibido la secuencia de correlación para obtener la media. Para

realizar esta operación el número de secuencias es potencia de 2 por lo que se realiza la división con un desplazamiento de bits. Este valor medio se almacena en Phase_RAM. Una vez se ha almacenado el valor medio de todas las fases, se pasa esta información a Phase_selector donde se decide la fase óptima. Calibration_control gestiona el control del bloque de calibración.

El correlador se sintetiza con un número de celdas igual a los bits de calibración. Pero el control cuenta tantas muestras por secuencia como el número de bits de la secuencia de calibración y unas muestras extra para tener en cuenta el retardo. El control empieza la cuenta de muestras cuando la señal valid_corr_data está activa, por lo que es necesario retrasarla un número de muestras igual a las muestras extra para incluir el retardo. En caso contrario, el control comienza a trabajar antes de que el correlador haya procesado la secuencia completa.

Para ello, en el bloque Delay se utiliza un buffer de tamaño SC-N. Se podría modificar el correlador como alternativa, pero eso conllevaría un consumo de elementos lógicos ligeramente superior.

Como parámetros configurables tiene todos los utilizados por los diferentes módulos:

- N_C: Número de secuencias de calibración por fase.
- N: Número de bits de la señal de correlación.
- C: Configuración de las celdas del correlador.
- W: Ancho de bits del dato de entrada.
- P_W: Ancho de bits de la fase.
- C_W: Ancho de bits de los datos del correlador.
- SC: Número de muestras contadas por secuencia.
- PH: Número de fases.

Como entradas:

- clk: Reloj del sistema.
- phase_change _done: Indica que se ha realizado el cambio de fase.
- i_data: datos de entrada.
- start_calibration: Señal de inicio de la fase de calibración.
- rst: Reset del sistema.

Las salidas son las siguientes:

- increment_ph: Señal que indica que se incrementa la fase.
- enable_tx: Habilitación del transmisor.
- calibration_end: Final de la fase de calibración.
- o_ph: Fase óptima.

2.3.1.1 Correlador

La función del correlador es comparar la señal recibida con una secuencia almacenada y detectar coincidencias. Este módulo se utiliza en la fase de calibración para determinar las fases más robustas. El bloque del cálculo del BER también utiliza un correlador para detectar la secuencia de calibración en la señal recibida.

El correlador realiza una convolución de la señal recibida con una señal fijada. La convolución consiste en desplazar una de las señales n muestras, multiplicar las muestras de ambas señales y sumar los resultados.

$$y(n) = x(n) * h(n) = \sum_{r=-\infty}^{\infty} h(r) \cdot x(n-r)$$

Ecuación 2.1 Convolución.

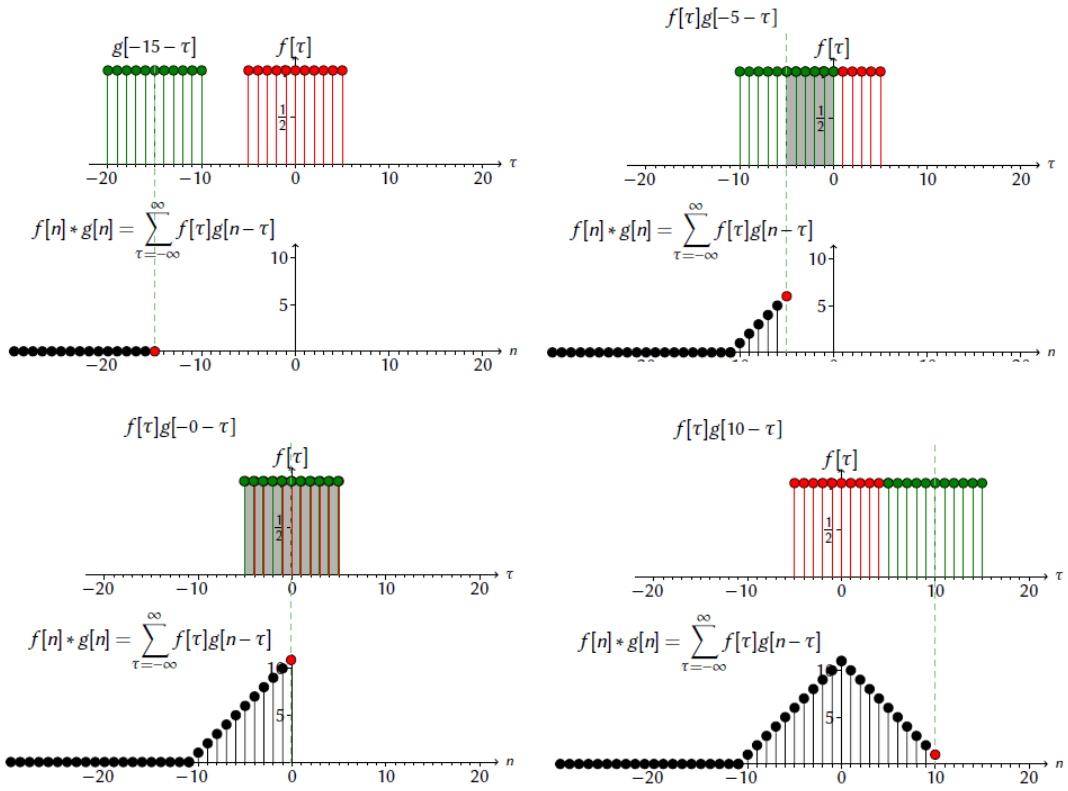


Figura 2.16 Visualización de la operación de convolución [8].

Cuando la señal recibida coincida con la señal fijada, el valor de la operación de convolución alcanzará un máximo. La señal fijada en el bloque del correlador es la secuencia de calibración. Con lo que es posible detectar dicha secuencia durante la transmisión de datos.

Para este caso en concreto, la señal recibida es una secuencia binaria comprendida entre 1 y -1. El resultado de la multiplicación de muestras iguales es 1, por lo que el valor máximo de correlación coincide con el número de bits de la secuencia de calibración.

La implementación digital de este módulo es muy similar a la de un filtro FIR, en este caso se utiliza una arquitectura paralela debido a las restricciones temporales del sistema.

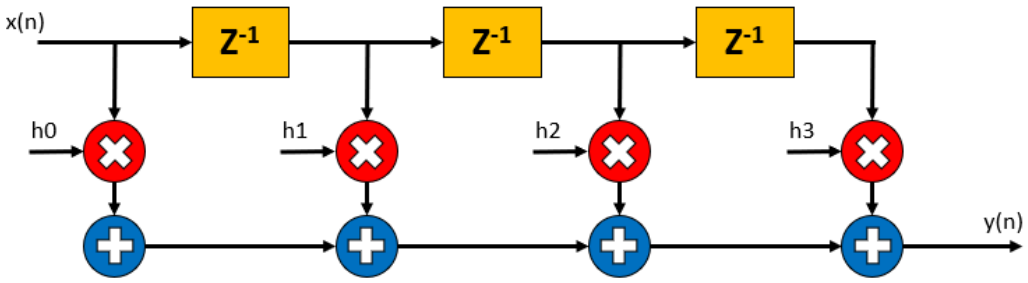


Figura 2.17 Arquitectura correlador.

Conociendo que la señal recibida es una secuencia de 1 y -1, es posible modificar la arquitectura para eliminar los multiplicadores del sistema. Cuando se compara una muestra recibida con un 1, se realiza la suma con el mismo valor de la muestra. Cuando se compara con un -1, se realiza la operación con complemento a dos de la muestra.

Para implementarlo en SystemVerilog, se descompone la operación en dos tipos de celdas distintas. Dependiendo de un parámetro C de entrada, se sintetiza un tipo de celda u otro tal y como se muestra en la Figura 2.18.

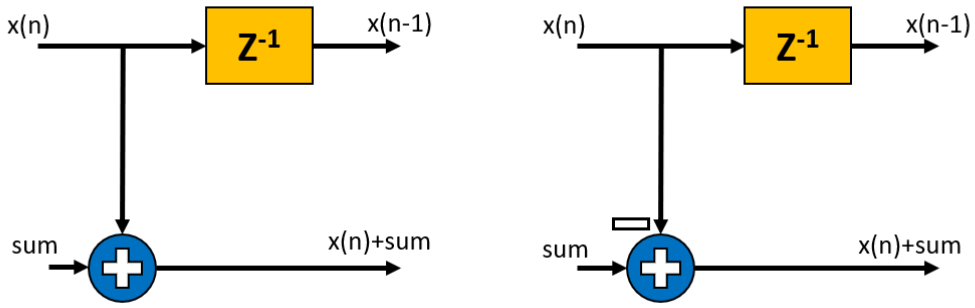


Figura 2.18 Izqda. celda coeficiente 1, Dcha. celda coeficiente -1.

El módulo tiene como parámetro de configuración la secuencia de calibración, que se utiliza para montar el sistema con las celdas correspondientes. En la Figura 2.19, se muestra un ejemplo de configuración del correlador.

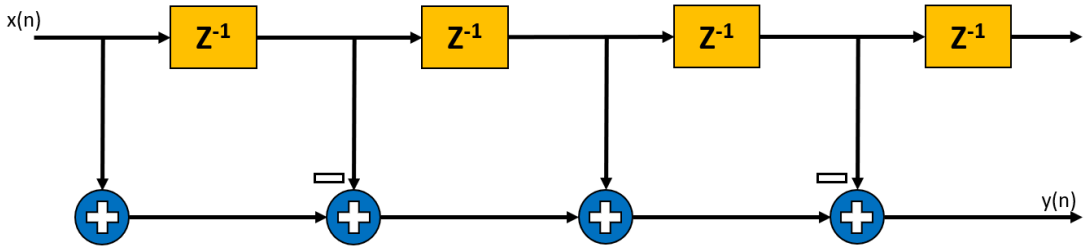


Figura 2.19 Correlador con secuencia de calibración 1, -1, 1, -1.

Para acortar el camino de la lógica combinacional, se realiza una segmentación del sistema. Se introduce un registro después de cada sumador y después de cada registro de la señal de entrada para sincronizar los tiempos.

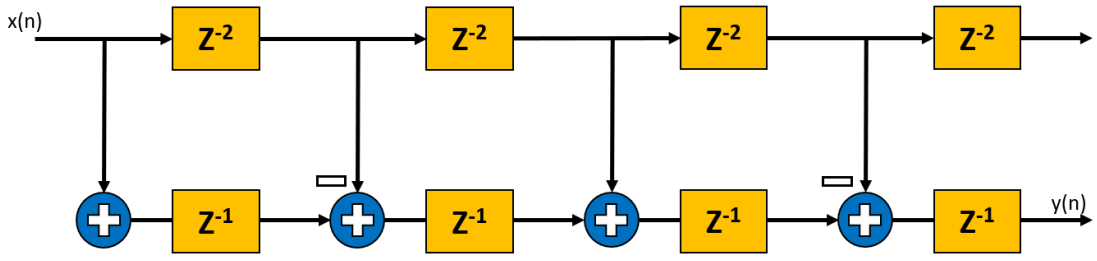


Figura 2.20 Correlador segmentado.

La celda cuenta con tres parámetros de configuración:

- C: Indica el valor del coeficiente de la celda. Cuando C=0 el coeficiente es -1, en caso contrario es 1.
- W: Es el número de bits del dato de entrada.
- G: Es el número de bits de guarda. El máximo de correlación corresponde con el número de bits de la señal de calibración. Así que los bits necesarios son el logaritmo en base 2 del número de bits de la señal de calibración, más un bit adicional de signo. Se realiza con la función $clog2.

En cuanto a las entradas y salidas de la celda:

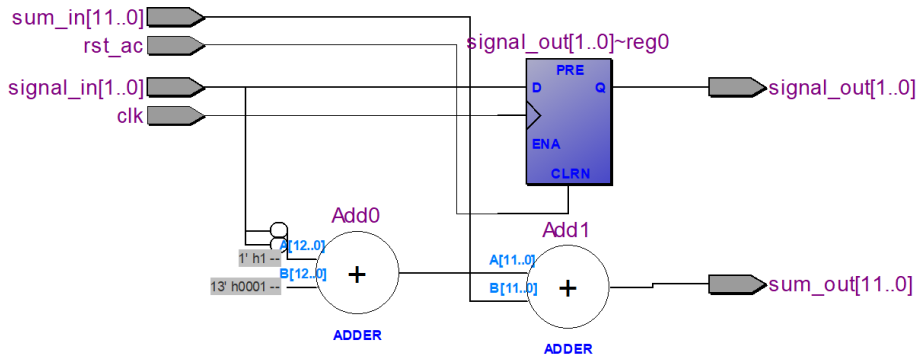
- signal_in: Señal de entrada, con formato [W-1:0].
- sum_in: Entrada con el resultado de la suma de la celda anterior, con formato [W+G-1:0].
- rst_ac: Reset asíncrono del sistema, activo a nivel alto.
- val_in: Entrada binaria que informa de una muestra válida en signal_in.
- clk: Entrada de reloj.
- ce: Entrada de habilitación de la celda.
- signal_out: Salida de la señal de entrada retrasada, con formato [W-1:0].
- val_out: Salida binaria que indica que hay una muestra válida en signal_out.
- sum_out: Salida con el resultado de la suma de la muestra a procesar con la suma anterior, con formato [G-1:0].

La señal de entrada se registra una vez dentro de la celda, y de nuevo a la salida para sincronizar las muestras de la señal con la suma registrada.

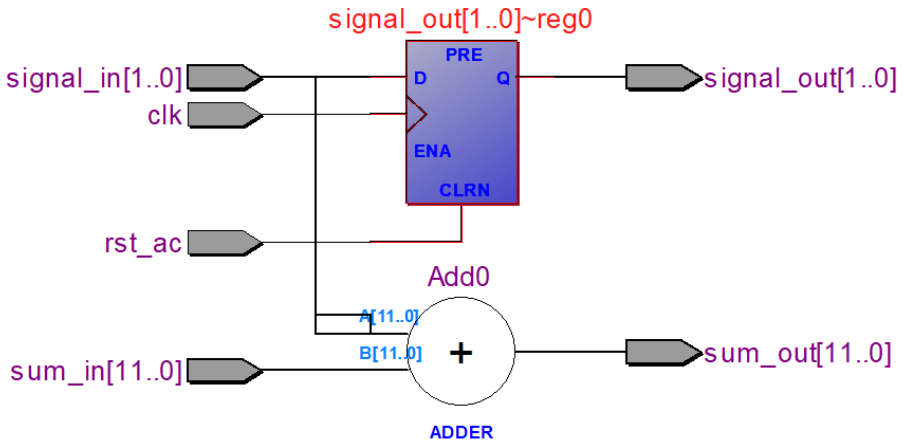
La señal val_out registra el estado de val_in, así estará activa cuando comience a obtenerse el valor de correlación.

La operación suma se sintetizará dependiendo del parámetro C. Cuando es 0, sum_out será el complemento a 2 de signal_in más sum_in. Si es distinto de 0, será signal_in más sum_in.

Para observar como el sintetizador implementa las celdas, se simplifica el código, eliminando los registros y las señales de validación. Obteniendo las celdas mostradas en la Figura 2.21.



(a)



(b)

Figura 2.21 Celdas del correlador. (a) Coeficiente -1. (b) Coeficiente 1.

Al compilar ambas celdas, los elementos lógicos utilizados coinciden.

Top-level Entity Name	CORRELATOR_CELL
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	11 / 41,910 (< 1 %)
Total registers	2
Total pins	49 / 499 (10 %)

Figura 2.22 Síntesis de ambos tipos de celda del correlador.

En la capa superior a la celda, se encuentra el módulo CORRELATOR. Cuenta con los siguientes Parámetros de configuración:

- N: Número de bits de la señal de calibración.
- C: Configuración de la señal de calibración. El número introducido en binario se utiliza como modelo para determinar qué tipo de celda se utiliza. Los unos representan un coeficiente 1 y los ceros un coeficiente -1. Por ejemplo, para la señal de calibración 1, -1, 1, -1; C sería 1010 en binario o 10 en decimal.
- W: Número de bits de la señal de entrada.

Las señales de entrada y salida del sistema son:

- i_data: Señal de entrada, con formato [W-1:0].
- rst_ac: Reset asíncrono del sistema, activo a nivel alto.
- val_in: Entrada binaria que informa de una muestra válida en i_data.
- clk: Entrada de reloj.
- ce: Entrada de habilitación de las celdas.
- o_data: Salida del sistema con el resultado de la correlación, con formato [W+ceil(log2(N))-1:0].
- val_out: Salida binaria que indica que hay una muestra válida en o_data.

2.3.1.2 Detector de picos de correlación

Es el módulo encargado de detectar los picos de correlación y calcular el valor medio de los máximos de correlación en cada fase. Estos máximos se acumulan con cada una de las transmisiones de la señal de calibración por fase y, finalmente, se calcula el valor medio. El número de secuencias se recibe como parámetro y el ancho de bits de los datos de entrada es parametrizable. Para poder realizar el cálculo del valor medio, el número de secuencias enviadas por fase debe ser potencia de 2. De esta forma puede realizarse la división como un desplazamiento de bits. Señales de control gestionan el flujo de funcionamiento del módulo.

El módulo utiliza 2 parámetros de configuración:

- C_W: Tamaño de los datos de entrada y salida.
- N_C: Número de secuencias enviadas por fase.

Además, se utilizan los parámetros n_desp y n_desp_w, que dependen de los dos anteriores:

- n_desp: Se utiliza para determinar el desplazamiento de bits del acumulador para obtener el valor medio. n=log2(N_C).
- n_desp_w: Se utiliza para determinar el ancho de bits del acumulador. El máximo del acumulador será igual a N_C multiplicado por el máximo que puede obtenerse con un ancho de bits C_W, luego n_desp_w=log2(N_C*2^C_W).

Respecto a las entradas y salidas del sistema:

- clk: Entrada de reloj.
- ce: Entrada de habilitación del detector de picos.
- acc_en: Entrada de habilitación del acumulador.
- acc_rst: Reset del acumulador.
- rst: Reset del sistema, activo a nivel alto.
- i_data: Entrada de datos. Recibe los datos del módulo correlador.
- o_data: Salida del sistema, valor medio del máximo de correlación para un número de secuencias de calibración determinado.

En la Figura 2.23, se muestra la parte de la máquina de estados que determina las señales de activación.

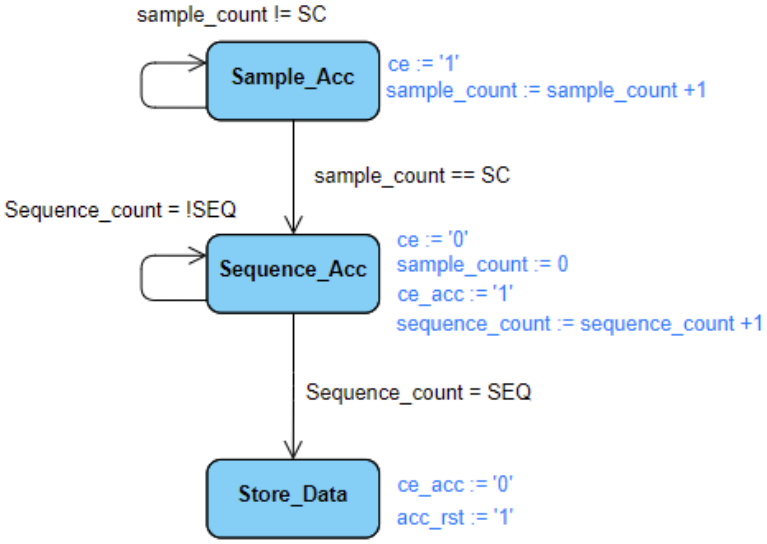


Figura 2.23 Máquina estados PEAK_DETECTOR.

Ce habilita el registro que almacena el pico máximo, una vez se han recibido todas las muestras de una secuencia, se activa el acumulador con ce_acc y se resetea el registro max_peak. Una vez recibidas todas las secuencias, se resetea el acumulador.

En la Figura 2.24 se muestra el esquema del módulo.

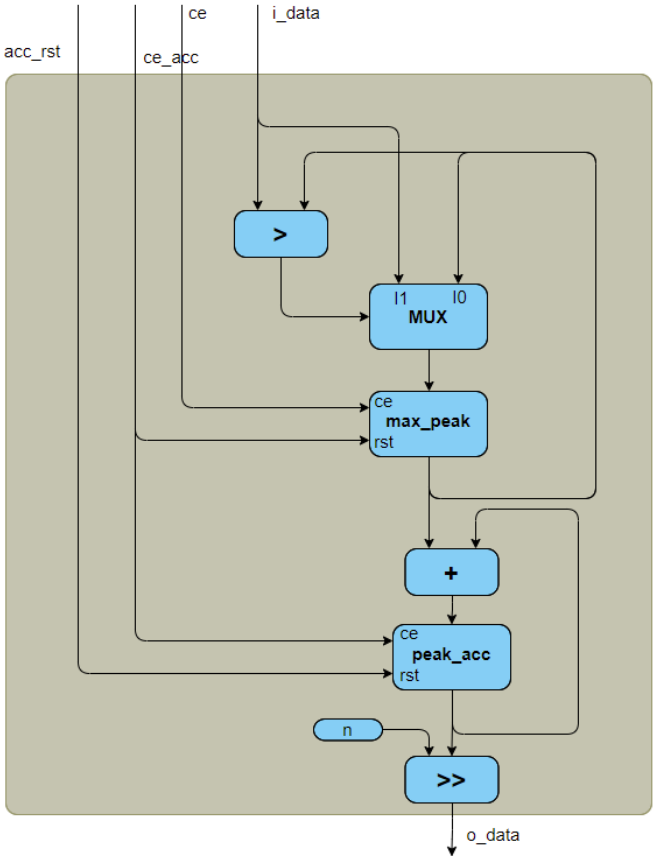


Figura 2.24 Esquema módulo PEAK_DETECTOR.

Se crea un registro max_peak de tamaño [C_W:0], que almacena el máximo de correlación para cada secuencia de calibración recibida. Se crea también el registro peak_accumulator de tamaño [n2:0] que almacena la suma del máximo de correlación de la secuencia actual con los máximos acumulados de las anteriores.

Cuando ce sea activo y el dato de entrada sea mayor que el valor registrado en max_peak, el dato de entrada se guardará en el registro. Ce es una señal de control que se activa mientras se recibe la secuencia de calibración.

Si acc_en es activo, significa que se ha finalizado la recepción de una secuencia de calibración. Al registro peak_accumulator se le suma el valor de max_peak y se resetea el registro max_peak.

La señal acc_rst indica que se ha finalizado el número de transmisiones para la fase actual, así que se resetea el registro peak_accumulator para comenzar a procesar los datos de la siguiente fase.

La salida del sistema es el valor del registro peak_accumulator desplazado n bits para calcular el valor medio.

2.3.1.3 Almacenamiento en RAM

Es el módulo encargado de almacenar los datos de valor medio de correlación en cada fase. Una vez finalizado el envío de señales de calibración, realiza un barrido de los datos almacenados. Debido al retraso por el medio entre el transmisor y el receptor, no se puede saber con seguridad en qué fase comienzan a procesarse los datos. Así que otra función de este módulo es la de enviar los valores ordenados al módulo encargado de determinar la fase óptima. En la Figura 2.25 se muestra el efecto de este desfase en la curva de correlación.

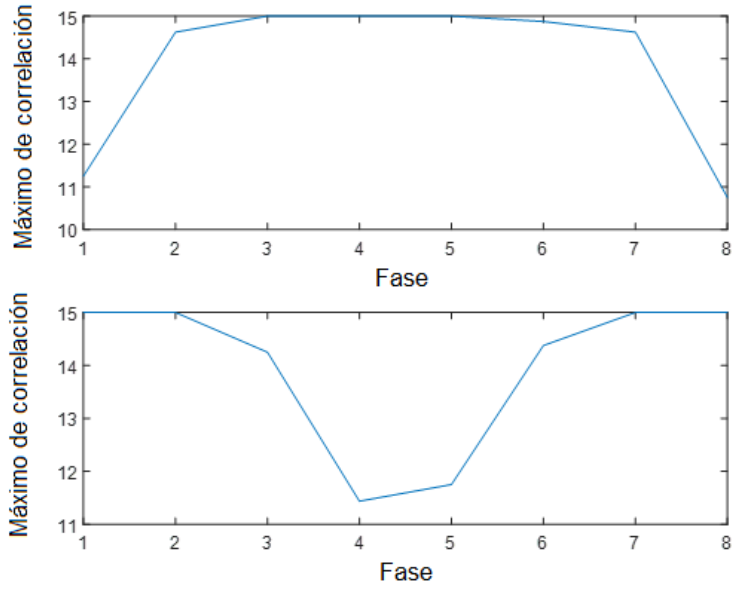


Figura 2.25 Curvas de correlación con 8 fases. Superior: curva de correlación, inferior: curva de correlación desfasada.

Se conoce la naturaleza de la curva de correlación de las fases gracias a las pruebas realizadas previamente en Matlab. Teniendo en cuenta esto y para simplificar el diseño del módulo que determina la fase óptima, PHASE_RAM envía los datos comenzando por la fase con menor correlación. Con los datos en la memoria, se calcula un umbral que determina qué fases son válidas. Se envía la fase con la correlación mínima y el umbral al siguiente módulo, ya que son datos necesarios para calcular la fase óptima.

El módulo utiliza 3 parámetros de configuración:

- C_W: Ancho de bits de los datos de correlación.
- P_W: Ancho de bits de los datos de fase.
- PH: Número de fases.

Respecto a las entradas y salidas del sistema:

- clk: Entrada de reloj.
- ce: Entrada de habilitación de escritura de la memoria.
- ce_phase_selector: Entrada de habilitación del barrido de la memoria.
- rst: Reset del sistema, activo a nivel alto.
- i_data: Entrada de datos. Recibe los datos con el valor medio de correlación.
- address: Entrada de datos con la fase en la que se está trabajando. Se utiliza como dirección de la memoria donde se almacenan los datos de i_data.
- o_data: Salida del sistema con los valores almacenados en la memoria.
- th: Salida con el umbral calculado.
- min_ph: Salida con la fase que tiene el mínimo de correlación.

La parte de la máquina de estados que gobierna las señales del módulo es la mostrada en la Figura 2.26.

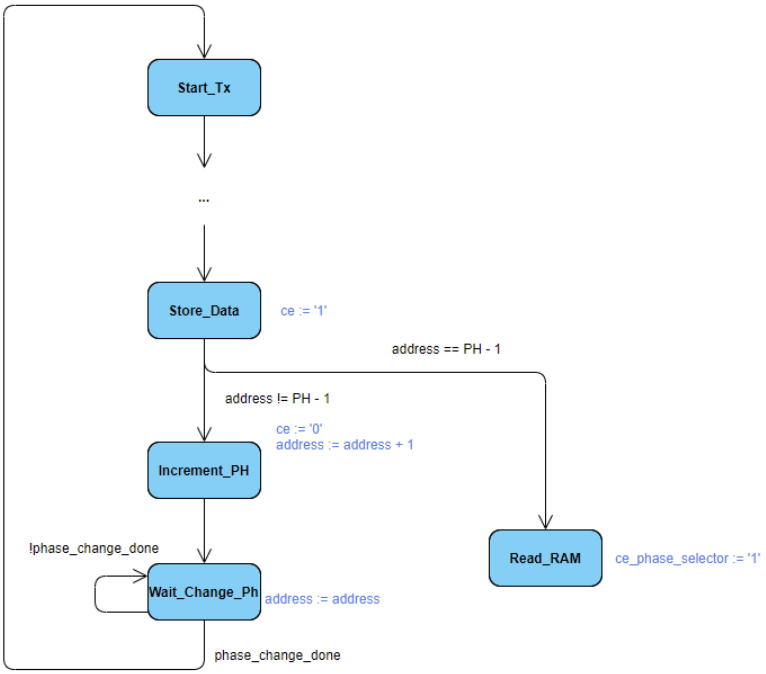


Figura 2.26 Máquina de estados PHASE_RAM.

La señal ce habilita la escritura de datos en la RAM en la posición address. Una vez guardado el dato se incrementa la dirección y se espera al cambio de fase. Si es la última fase, se habilita la señal ce_phase_selector que habilita el barrido de datos de la memoria comenzando por la posición RAM_SWEEP.

En la Figura 2.27 se muestra el esquema del módulo.

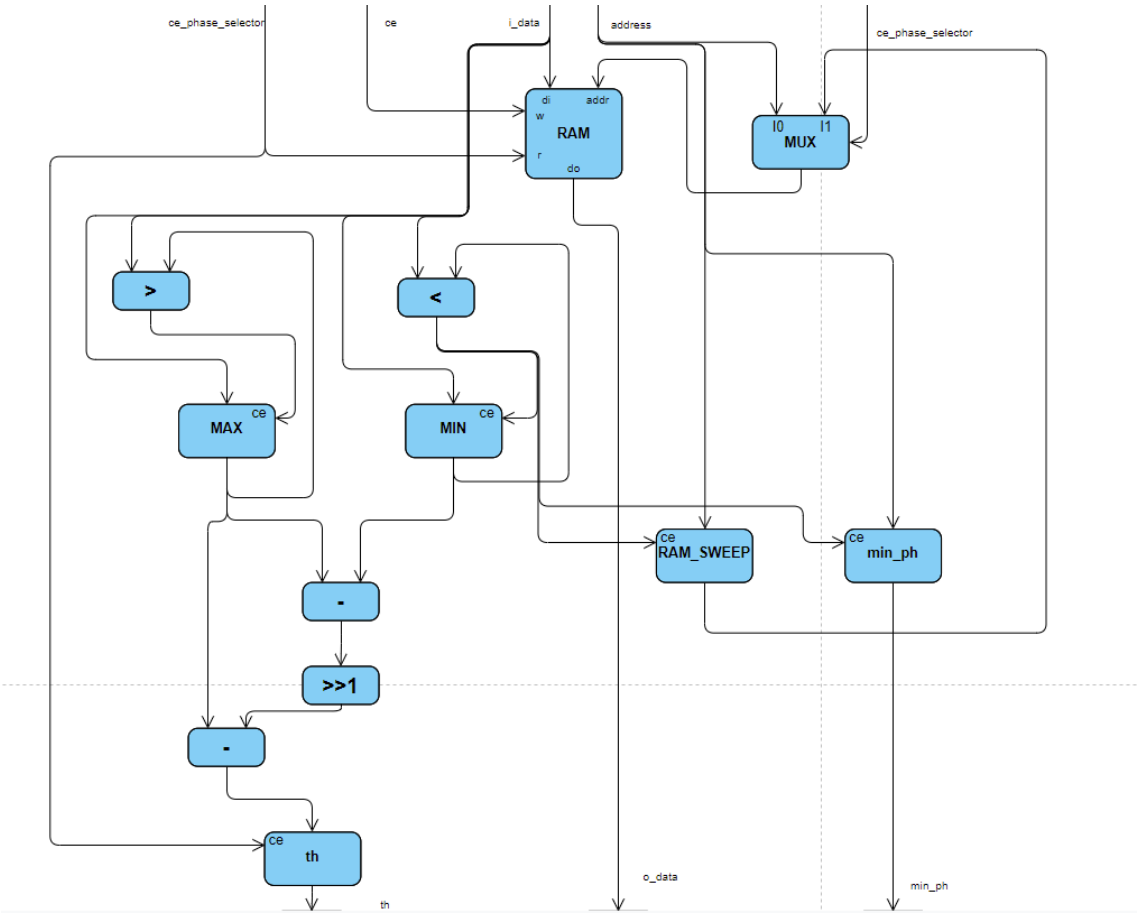


Figura 2.27 Esquema PHASE_RAM.

Se crean los registros MAX y MIN de tamaño [C_W:0] que almacenan el valor máximo y mínimo de correlación. RAM_SWEEP es un registro de tamaño [P_W:0] que almacena la fase en la que la correlación es mínima. Esta es la dirección de memoria que comienza a leerse al enviar los datos al siguiente módulo. RAM es una memoria de [PH-1:0] direcciones de tamaño [C_W:0] donde se almacenan los datos recibidos.

La señal de control ce es activa cuando se han procesado todas las secuencias de calibración de una fase. Cuando esto ocurre se almacena el dato con la media de correlación recibida mediante i_data en la memoria RAM en la posición determinada por address, que va por orden de fase desde la 0 hasta PH-1.

Concurrentemente se implementa la lógica para almacenar el máximo de correlación en el registro MAX, y el mínimo en MIN. Así como la fase con el mínimo de correlación en RAM_SWEEP y min_ph.

La señal de control ce_phase_selector es activa cuando se ha finalizado el procesado de todas las fases. Cuando esto ocurre el módulo realiza un barrido de las posiciones de memoria comenzando por RAM_SWEEP y envía los datos almacenados a través de la salida o_data al módulo que determina la fase óptima. Además, se realiza el cálculo del umbral utilizado para decidir si la fase es válida. El umbral se calcula como la diferencia entre el valor almacenado en el registro MAX y la resta entre el registro MAX y MIN desplazado un bit.

$$\mathrm{th} = \mathrm{MAX} - ((\mathrm{MAX} - \mathrm{MIN}) >> 1)$$

Ecuación 2.2 Cálculo del umbral.

2.3.1.4 Selección de fase

Este módulo se encarga de seleccionar la fase óptima a partir de los datos facilitados por el bloque PHASE_RAM. La fase óptima se determina como la fase intermedia entre todas las que superan el umbral. Una vez obtenida la fase óptima, se realiza el cálculo para compensar el offset entre la curva ordenada que recibe del bloque PHASE_RAM y la curva original.

El módulo renumera las fases para hacer el cálculo según se muestra en la Figura 2.28. Por eso es necesario realizar posteriormente el cálculo para seleccionar la fase según la curva recibida. En el caso de la imagen, la fase seleccionada inicialmente por PHASE_SELECTOR sería la 4, que corresponde originalmente con la fase 8. El cálculo final se encarga de hacer la compensación y obtener la fase 8 como fase elegida.

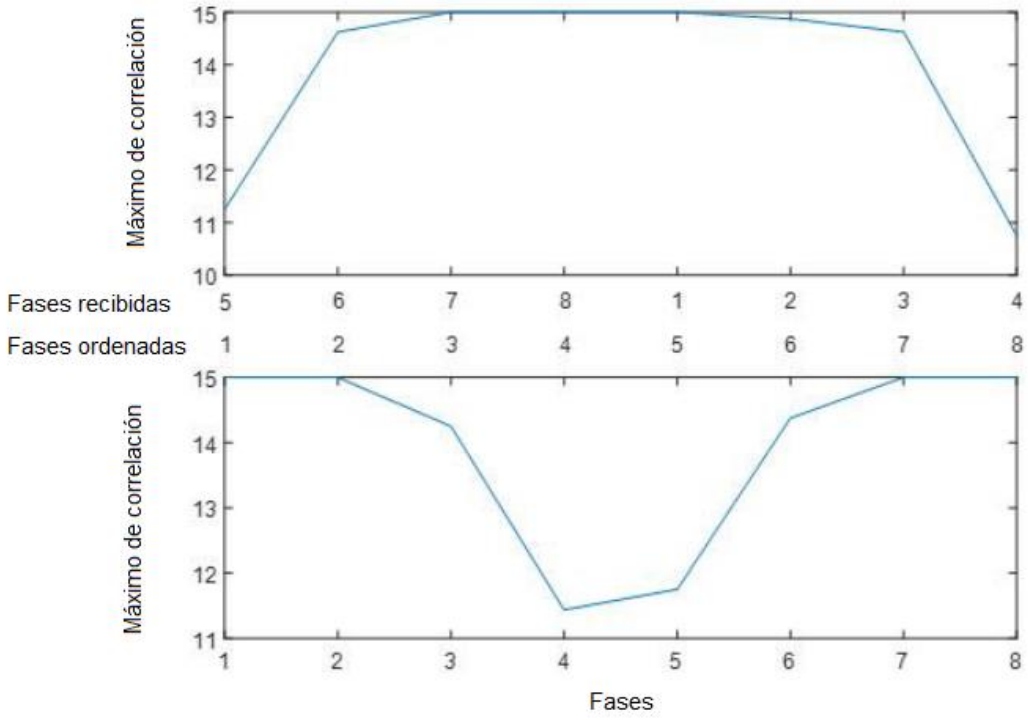


Figura 2.28 superior: curva de correlación ordenada. (fases recibidas: numeradas según se reciben del transmisor, fases ordenadas, según el módulo PHASE_RAM); inferior: curva de correlación recibida.

El módulo utiliza 3 parámetros de configuración:

- C_W: Ancho de bits de los datos de correlación.
- P_W: Ancho de bits de los datos de fase.
- PH: Número de fases.

Respecto a las entradas y salidas del sistema:

- clk: Entrada de reloj.
- ce: Entrada de habilitación del proceso de selección de fase.
- ce_offset_adjust: Entrada del cálculo de la fase desfasada.
- rst: Reset del sistema, activo a nivel alto.
- i_corr: Entrada de datos. Recibe los datos con el valor medio de correlación.
- th: Entrada de datos con el umbral a partil del cual la fase es considerada válida.
- Min_ph: Entrada con la fase correspondiente al mínimo de correlación. Se usa para calcular el offset.
- o_ph: Salida con la fase óptima.

La Figura 2.29 muestra la parte de la máquina de estados con las señales de control del módulo.

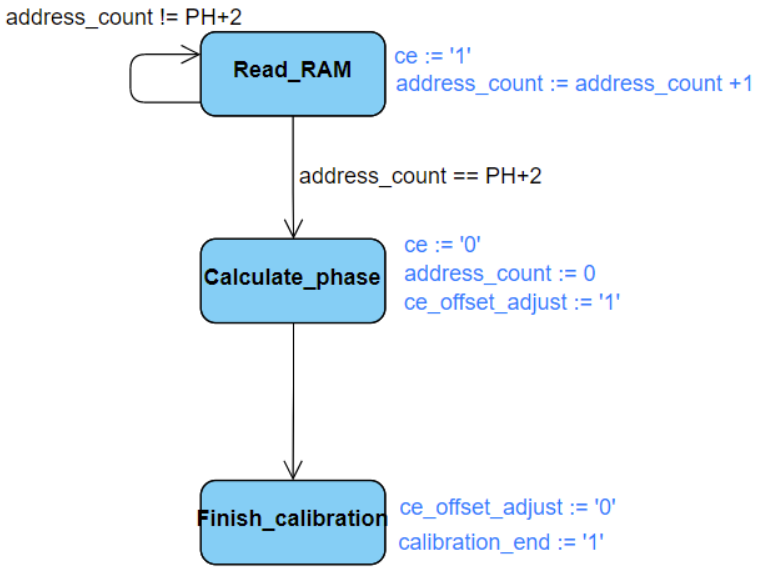


Figura 2.29 Máquina de estados PHASE_SELECTOR.

Cuando se reciben todas las muestras de calibración, Read_RAM activa la señal ce. Esta señal habilita los registros que forman parte del sistema encargado de decidir la fase óptima. Está activa durante 2 muestras más que el número de fases porque habilita también el módulo anterior PHASE_RAM, y este tiene 1 retardo entre la entrada y la salida. El retardo extra es para actualizar el registro encargado de realizar el cálculo del offset que se explica más adelante.

Una vez se conoce la fase óptima, la señal ce_offset_adjust se activa y habilita el registro encargado de almacenar el resultado del cálculo del offset.

Tras un ciclo de reloj se deshabilita la ce_offset_adjust y se envía la señal de fin de calibración.

La Figura 2.30 muestra el esquema del módulo PHASE_SELECTOR.

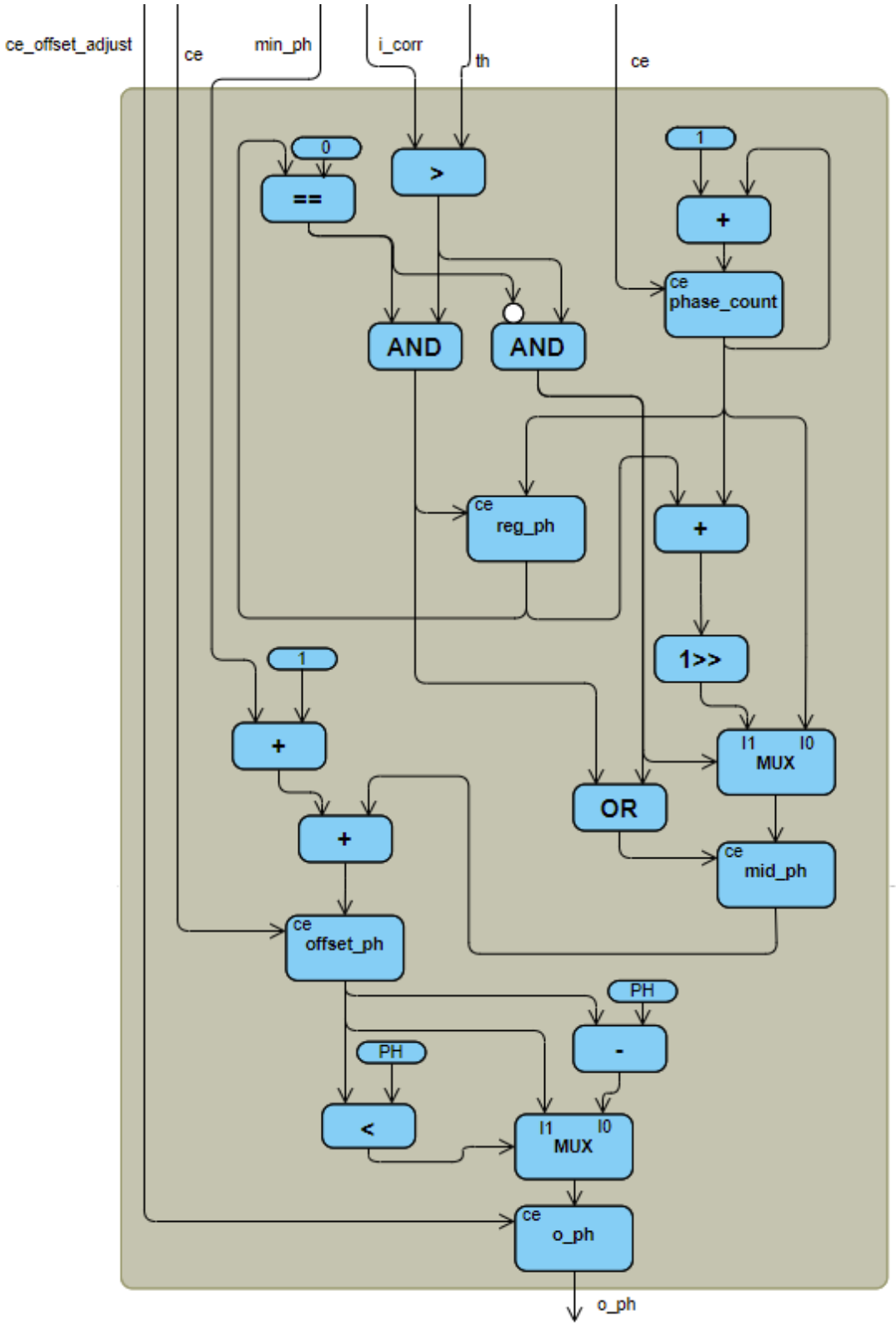


Figura 2.30 Esquema PHASE_SELECTOR.

El registro phase_count es un contador que asigna fases temporales a los valores de correlación medios recibidos desde PHASE_RAM. Reg_ph registra la primera fase que supera el umbral de correlación. Mid_ph es un registro que almacena la fase central entre las que superan el umbral. Offset_ph se utiliza para compensar el offset.

La misma señal de control activa el módulo PHASE_RAM y PHASE_SELECTOR. Hay un retardo entre la activación y la salida del módulo PHASE_RAM, por lo que la primera muestra válida es la correspondiente a phase_count = 1.

Cuando la señal de control ce está activa, comienza la obtención de la fase óptima. Esta es la fase central de entre todas las fases cuyo máximo de correlación supera el umbral th. Phase_count se incrementa con cada muestra recibida y si el valor de correlación supera el umbral, se registra el valor del contador en reg_ph. Para cada nueva muestra que llegue que supere el umbral, se asigna a mid_ph el valor medio entre reg_ph y la fase asignada a la nueva muestra. Esto se consigue sumando reg_ph + phase_count y desplazando el resultado un bit.

Concurrentemente offset_ph es la fase central más la fase con el valor mínimo de correlación min_ph recibido como valor de entrada del módulo. A min_ph se le suma uno. Esto se hace porque en este módulo el rango de la fase está entre 1 y PH mientras que anteriormente estaba entre 0 y PH-1. Por lo que es necesario incrementar min_ph para ajustarlo al nuevo rango.

Cuando se han leído todos los valores de memoria, desde el control se activa la señal ce_offset_adjust y comienza el ajuste de offset. Se comprueba el valor de offset_ph, si es menor que el número de fases, offset_ph corresponde a la fase óptima. En caso contrario, la fase óptima es offset_ph menos el número de fases.

2.3.1.5 Control

Módulo encargado del control de la fase de calibración. El control se implementa como una máquina de estados de Moore en el que cada estado proporciona las señales de control de los distintos módulos del bloque de calibración. Además, debe enviar señales para iniciar o parar el transmisor y tiene una señal de salida para indicar el final del proceso de calibración. La transmisión es continua para todas las secuencias de una fase.

La primera etapa inicial Init_State pondrá todos los contadores y señales a 0. En cuanto llegue la señal de inicio de calibración, pasará a la siguiente etapa Start_Tx, donde se activará el transmisor y el correlador. El sistema se mantendrá en esta etapa hasta que empiecen a salir muestras válidas del correlador.

A continuación, en Sample_acc, se incrementará un contador por cada muestra de entrada y se activará el detector de picos. Seguirá incrementándose hasta que cuente SC muestras, que son las de la señal de calibración más alguna muestra teniendo en cuenta el retardo de transmisión.

Cuando el contador llega a SC significa que se ha recibido la primera señal de calibración así que se pasa al estado Sequence_Acc, donde se cuentan las secuencias recibidas. En este estado se activa el acumulador del detector de picos y se incrementa el contador de secuencias. Si el número de secuencias es inferior a SEQ-1 se vuelve al estado anterior para contar las muestras de la nueva señal. Si es SEQ-1, se pasa al estado Stop_Tx. Y si es SEQ se pasa al estado Store_Data.

En Stop_Tx, se para la transmisión de datos. Se hace cuando el contador está en SEQ-1 porque el sistema empieza a contar cuando la primera muestra sale del correlador y, en este momento, la primera secuencia ya ha sido enviada. Para evitar un retraso de una muestra en la penúltima secuencia, cuando el sistema pasa por este estado, se comporta como Sample_Acc. Se activa el detector de picos y el acumulador de muestras.

Store_Data actúa cuando todas las secuencias de calibración de una fase han sido recibidas. Envía la señal al bloque PHASE_RAM, para que almacene el valor medio del máximo de correlación de la fase actual. Además, se envía la señal a PEAK_DETECTOR para borrar el acumulador. Si el contador de fase es igual a PH - 1, significa que el sistema está procesando la última fase y en lugar de pasar a Increment_Phase, la siguiente etapa será Read_RAM.

En Increment_Phase, se incrementa el contador con el valor de la fase y se envía la señal de cambio de fase al bloque encargado de realizar esta función. Se necesita este estado extra porque la memoria en PHASE_RAM utiliza como dirección este contador, y si se incrementa en el mismo

estado que se envía la señal de control al bloque, se almacena el dato en la posición de memoria siguiente. Por ejemplo, el dato de la primera fase se almacenaría en la posición de memoria 1 en lugar de la 0.

Tras un ciclo de reloj pasa a la siguiente fase Wait_Change_Ph, que espera a recibir la señal indicando que el bloque encargado de realizar el cambio de fase ha finalizado con su labor. Cuando llega la señal de cambio de fase, se vuelve a la etapa Start_Tx donde se empieza a transmitir las señales de calibración para la siguiente fase.

En Read_RAM, se envía la señal a PHASE_RAM para que realice un barrido de la memoria y se activa el bloque PHASE_SELECTOR. Se mantendrán estas señales activas con un contador durante PH+2 muestras. +2 porque hay un retardo entre la entrada y salida de PHASE_RAM cuando comienza el barrido, y otro en PHASE_SELECTOR para realizar el cálculo de la fase óptima. Una vez llega el contador al valor indicado, se pasa a la etapa Calculate_Phase.

En Calculate_Phase se envía la señal al bloque PHASE_SELECTOR para realizar el cálculo del offset.

Tras un ciclo de reloj evoluciona a la etapa final Finish_Calibration, donde se envía la señal indicando que se ha finalizado el proceso de calibración. Tras un ciclo de reloj se vuelve a la etapa inicial.

En la Figura 2.31 se muestran las etapas del sistema y las condiciones para pasar de una a otra.

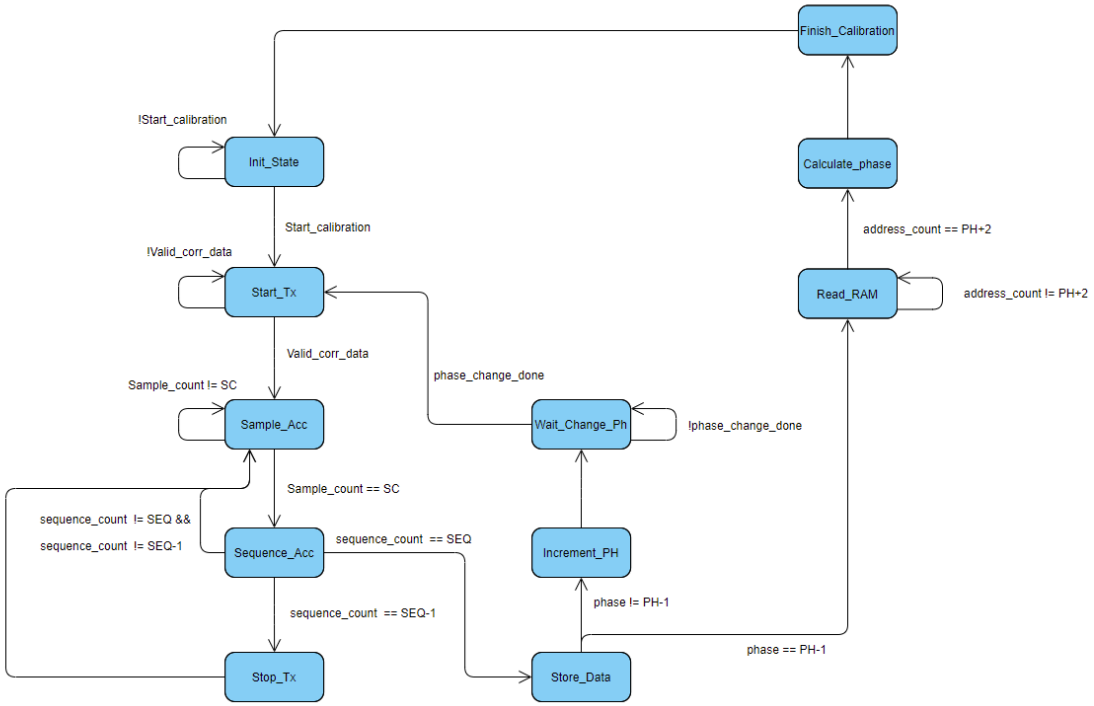


Figura 2.31 Diagrama de flujo del control.

El módulo utiliza 4 parámetros de configuración:

- P_W: Ancho de bits de los datos relacionados con la fase.
- SC: Número de muestras de la señal de calibración más un extra para incluir el retardo.
- PH: Número de fases.
- SEQ: Número de secuencias recibidas por fase.

Las señales de entrada del sistema son:

- start_calibration: Viene del control de nivel superior, indica que se comienza el proceso de calibración.
- valid_corr_data: Señal que se habilita desde el correlador cuando comienza a generar muestras válidas.
- phase_change_done: Indica que se ha cambiado la fase.
- clk: Reloj del sistema.
- rst: Señal de reset.

En cuanto a las salidas:

- ce_correlator: Habilitación del correlador.
- ce_peak_detector: Habilita el detector de picos.
- ce_peak_accumulator: Habilita el acumulador del detector de picos.
- ce_phase_ram_store: Habilita la escritura de la RAM, además es la señal que indica que se comienza con el cambio de fase.
- ce_phase_selector: Habilita la lectura de datos de la memoria y el selector de fase.
- ce_offset_adjust: Habilita el cálculo del ajuste de offset.
- calibration_end: Indica que se ha finalizado la calibración.
- enable_tx: Habilita la transmisión de datos.
- phase: Indica en qué fase se está trabajando, sirve como dirección para el bloque PHASE_RAM.

Se utilizan 3 contadores:

- sample_count: Contador de muestras. De tamaño [log2(SC):0].
- sequence_count: Contador de secuencias. De tamaño [log2(SEQ):0].
- address_count: Contador para mantener active el selector de fase durante PH+2 muestras. De tamaño [P_W:0].

En la Tabla 2.1, se muestran las señales que se activan en cada etapa. Si no se especifica en la etapa, la señal mantiene el valor de su etapa anterior.

Init_State		Todo a 0	
Start_Tx		enable_tx = 1 ce_correlator = 1	
Sample_Acc		ce_peak_detector = 1 sample_count = sample_count + 1	
Sequence_Acc		ce_peak_detector = 0 ce_peak_accumulator = 1 sample_count = 0 sequence_count = sequence_count + 1	
Stop_Tx	ce_peak_detector=1 ce_peak_accumulator = 0 sample_count = sample_count + 1 enable_tx = 0	**Store_Data**	ce_correlator = 0 ce_peak_accumulator = 0 ce_phase_ram_store = 1 sequence_count = 0
Increment_Phase		ce_correlator = 0 ce_peak_accumulator = 0 ce_phase_ram_store = 0 sequence_count = 0 phase = phase + 1 increment_ph =1	
Wait_Change_Ph		Todo a 0	
Read_RAM		ce_phase_selector = 1 address_count = address_count + 1	

Calculate_Phase	ce_phase_selector = 0 ce_offset_adjust = 1 address_count = 0
Finish_Calibration	ce_offset_adjust = 0 calibration_end = 1

Tabla 2.1 Activación de señales en cada etapa de CALIBRATION_CONTROL.

2.3.1.6 Verificación del bloque de calibración

En Matlab se crea una nueva matriz que contenga los datos de las secuencias de calibración tras el filtro y el ruido ordenados por fase. Si se envían 10 secuencias, las primeras 10 filas corresponden a las 10 secuencias de calibración de la fase 1, las siguientes 10 a las de la fase 2 y así sucesivamente hasta completar todas las fases.

A la matriz se le añade el número de ceros definido en la cabecera para simular el retardo y mantenerlo constante durante todas las secuencias y se exporta en un fichero de texto como entrada de datos del test en ModelSim.

Finalmente, se crea un paquete con todos los parámetros que se invoca desde el test.

En ModelSim se genera un banco de pruebas para testear el hardware. En el bloque initial se abren los ficheros de texto y se inicializan las señales de entrada, se resetea el sistema y se habilita la señal de comienzo de calibración.

Cuando se activa la señal de cambio de fase, se activa 2 muestras después la señal de cambio de fase realizado. Se finaliza el test al activarse la señal calibration_end, y se hace la comprobación de la fase obtenida en hardware con la de Matlab. Tras realizar la simulación, se muestra en pantalla si los resultados de ambos modelos coinciden.

Se realiza una serie de test para comprobar que el sistema de calibración funciona bajo diferentes condiciones.

Variaciones en el retardo de transmisión:

Se comprueba el sistema con unos parámetros fijos, pero variando el retardo de transmisión.

- Polinomio secuencia de calibración [4 3 0]
- Fases: 16
- Snr dB: 15
- Transmisiones por fase: 32
- Muestras por secuencia de calibración: 19

Tx Delay: 0	Optimal phase is 9 Number of checked samples 10240
Tx Delay: 1	Optimal phase is 10 Number of checked samples 10240
Tx Delay: 4	Optimal phase is 13 Number of checked samples 10240
Tx Delay: 8	Optimal phase is 1 Number of checked samples 10240
Tx Delay: 12	Optimal phase is 5 Number of checked samples 10240
Tx Delay: 15	Optimal phase is 8 Number of checked samples 10240

Tabla 2.2 Test variando el retardo por transmisión.

Variaciones en el número de fases:

Se prueba con unos parámetros fijos, pero variando el número de fases.

- Polinomio secuencia de calibración [4 3 0]
- Tx Delay: 2
- Snr dB: 15
- Transmisiones por fase: 32
- Muestras por secuencia de calibración: 19

Fases: 8	Optimal phase is 7 Number of checked samples 5120
Fases: 16	Optimal phase is 11 Number of checked samples 10240
Fases: 21	Optimal phase is 14 Number of checked samples 13440
Fases: 32	Optimal phase is 19 Number of checked samples 20480
Fases: 64	Optimal phase is 35 Number of checked samples 40960

Tabla 2.3 Test variando el número de fases.

Variaciones en la secuencia de calibración:

Se prueba con unos parámetros fijos, pero variando la secuencia de calibración.

- Fases: 16
- Tx Delay: 2
- Snr dB: 15
- Transmisiones por fase: 32
- Muestras por secuencia de calibración: 19

Polinomio secuencia de calibración [3 2 0]	Optimal phase is 11 Number of checked samples 6144
Polinomio secuencia de calibración [4 3 0]	Optimal phase is 11 Number of checked samples 10240
Polinomio secuencia de calibración [5 4 0]	Optimal phase is 11 Number of checked samples 18432
Polinomio secuencia de calibración [6 5 0]	Optimal phase is 11 Number of checked samples 34816
Polinomio secuencia de calibración [7 6 0]	Optimal phase is 11 Number of checked samples 67584

Tabla 2.4 Test variando la secuencia de calibración.

Variaciones en el número de transmisiones por fase:

Se prueba con unos parámetros fijos, pero variando las transmisiones por fase.

- Polinomio secuencia de calibración [4 3 0]
- Fases: 16
- Snr dB: 15
- Tx delay: 2
- Muestras por secuencia de calibración: 19

Transmisiones por fase:8	Optimal phase is 11 Number of checked samples 2560
Transmisiones por fase: 32	Optimal phase is 11 Number of checked samples 10240
Transmisiones por fase: 128	Optimal phase is 11 Number of checked samples 40960

Transmisiones por fase: 256	Optimal phase is 11 Number of checked samples 81920
Transmisiones por fase: 1024	Optimal phase is 11 Number of checked samples 327680

Tabla 2.5 Test variando el número de secuencias transmitidas por fase.

Variaciones en ruido:

Se prueba con unos parámetros fijos, pero variando el ruido. En este caso el valor máximo de la media de correlación disminuirá para señales con más ruido.

- Polinomio secuencia de calibración [4 3 0]
- Fases: 16
- Transmisiones por fase: 32
- Tx Delay: 2
- Muestras por secuencia de calibración: 19

Snr dB: 2	Optimal phase is 11 Number of checked samples 10240 /CALIBRATION_BLOCK_TB/CALIBRATION_BLOCK_UUT/PHASE_RAM/MAX 13 /CALIBRATION_BLOCK_TB/CALIBRATION_BLOCK_UUT/PHASE_RAM/MIN 7
Snr dB: 5	Optimal phase is 11 Number of checked samples 10240 /CALIBRATION_BLOCK_TB/CALIBRATION_BLOCK_UUT/PHASE_RAM/MAX 14 /CALIBRATION_BLOCK_TB/CALIBRATION_BLOCK_UUT/PHASE_RAM/MIN 8
Snr dB: 8	Optimal phase is 11 Number of checked samples 10240 /CALIBRATION_BLOCK_TB/CALIBRATION_BLOCK_UUT/PHASE_RAM/MAX 15 /CALIBRATION_BLOCK_TB/CALIBRATION_BLOCK_UUT/PHASE_RAM/MIN 10
Snr dB: 12	Optimal phase is 11 Number of checked samples 10240 /CALIBRATION_BLOCK_TB/CALIBRATION_BLOCK_UUT/PHASE_RAM/MAX 15 /CALIBRATION_BLOCK_TB/CALIBRATION_BLOCK_UUT/PHASE_RAM/MIN 10
Snr dB: 14	Optimal phase is 11 Number of checked samples 10240 /CALIBRATION_BLOCK_TB/CALIBRATION_BLOCK_UUT/PHASE_RAM/MAX 15 /CALIBRATION_BLOCK_TB/CALIBRATION_BLOCK_UUT/PHASE_RAM/MIN 10

Tabla 2.6 Test variando el ruido.

Variaciones en el número de muestras por secuencia de calibración:

Se prueba con unos parámetros fijos, pero variando el número de muestras de la señal de calibración para incluir el retardo.

- Polinomio secuencia de calibración [4 3 0]
- Fases: 16
- Snr dB: 15
- Transmisiones por fase: 32
- Tx Delay: 2

Muestras por secuencia de calibración: 15	Optimal phase is 11 Number of checked samples 8224

Muestras por secuencia de calibración: 16	Optimal phase is 11 Number of checked samples 8704
Muestras por secuencia de calibración: 20	Optimal phase is 11 Number of checked samples 10752
Muestras por secuencia de calibración: 25	Optimal phase is 11 Number of checked samples 13312
Muestras por secuencia de calibración: 40	Optimal phase is 11 Number of checked samples 20992

Tabla 2.7 Test variando el número de muestras por secuencia de calibración.

Variaciones en múltiples parámetros:

A continuación, se observan casos de test en los que múltiples parámetros son distintos a los usados anteriormente.

Polinomio secuencia de calibración [4 3 0] Fases: 8 Snr dB: 3 Transmisiones por fase: 8 Muestras por secuencia de calibración: 17 Tx Delay: 0	Optimal phase is 5 Number of checked samples 1152 Pocas fases, pocas transmisiones y señal ruidosa.
Polinomio secuencia de calibración [5 4 0] Fases: 16 Snr dB: 10 Transmisiones por fase: 32 Muestras por secuencia de calibración: 36 Tx Delay: 10	Optimal phase is 3 Number of checked samples 21312 Número de secuencias no es potencia de 2.
Polinomio secuencia de calibración [4 3 0] Fases: 121 Snr dB: 10 Transmisiones por fase: 32 Muestras por secuencia de calibración: 22 Tx Delay: 102	Optimal phase is 43 Number of checked samples 89056 Alto número de fases y delay.
Polinomio secuencia de calibración [6 5 0] Fases: 20 Snr dB: 15 Transmisiones por fase: 1024 Muestras por secuencia de calibración: 73 Tx Delay: 7	Optimal phase is 18 Number of checked samples 1515520 Caso similar a una configuración real.

Polinomio secuencia de calibración [4 3 0] Fases: 15 Snr dB: 8 Transmisiones por fase: 32 Muestras por secuencia de calibración: 19 Tx Delay: 0	Optimal phase is 8 Number of checked samples 9600 Wn filtro 1/20
Polinomio secuencia de calibración [4 3 0] Fases: 15 Snr dB: 8 Transmisiones por fase: 32 Muestras por secuencia de calibración: 19 Tx Delay: 0	Optimal phase is 9 Number of checked samples 9600 Wn filtro 1/2

Tabla 2.8 Diferentes casos de test.

2.3.2 Cálculo del BER

Este bloque se encarga de realizar el cálculo del BER de la secuencia transmitida. El sistema comienza a funcionar cuando se recibe la señal de inicio de proceso de cálculo del BER. La señal recibida desde el transmisor está compuesta por una cabecera que indica el comienzo de la transmisión y la transmisión válida. Cuando se detecta la cabecera, comienza el cálculo del BER Se realiza comparando la señal de entrada con un LFSR interno. Este LFSR comparte la misma configuración que el LFSR utilizado por el transmisor para generar la señal. El bloque decide la habilitación del transmisor. En la Figura 2.32 se muestra el diagrama de flujo del sistema.

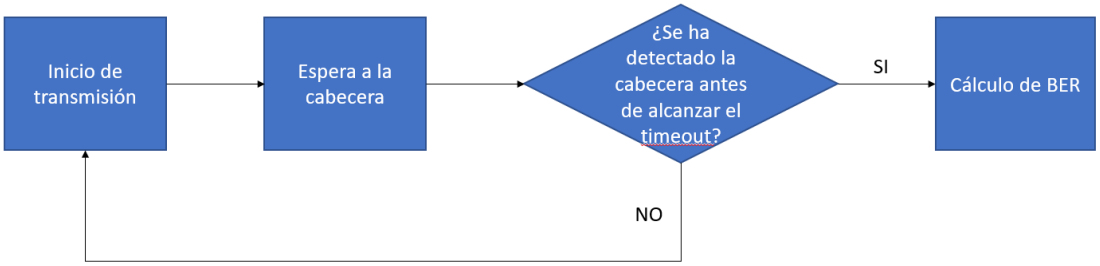


Figura 2.32 Diagrama de flujo proceso cálculo del BER.

La señal del transmisor se recibe en HEADER_DETECTOR donde espera a detectar la cabecera que indica el inicio de la transmisión. Este módulo introduce un retardo propio, además del retardo provocado por el correlador instanciado. Así que la señal que indica la detección de la cabecera se recibe con un delay igual al número de bits de la señal de cabecera más uno. Por ello se introduce un delay a la señal de entrada de N+1. Esta señal retrasada es la que se procesa en el módulo BER_CALC donde se compara con los datos generados por el LFSR para realizar el cálculo del BER. En la Figura 2.33 Esquema conexiones bloque BER_BLOCK. se muestra el esquema con las conexiones del bloque de cálculo del BER.

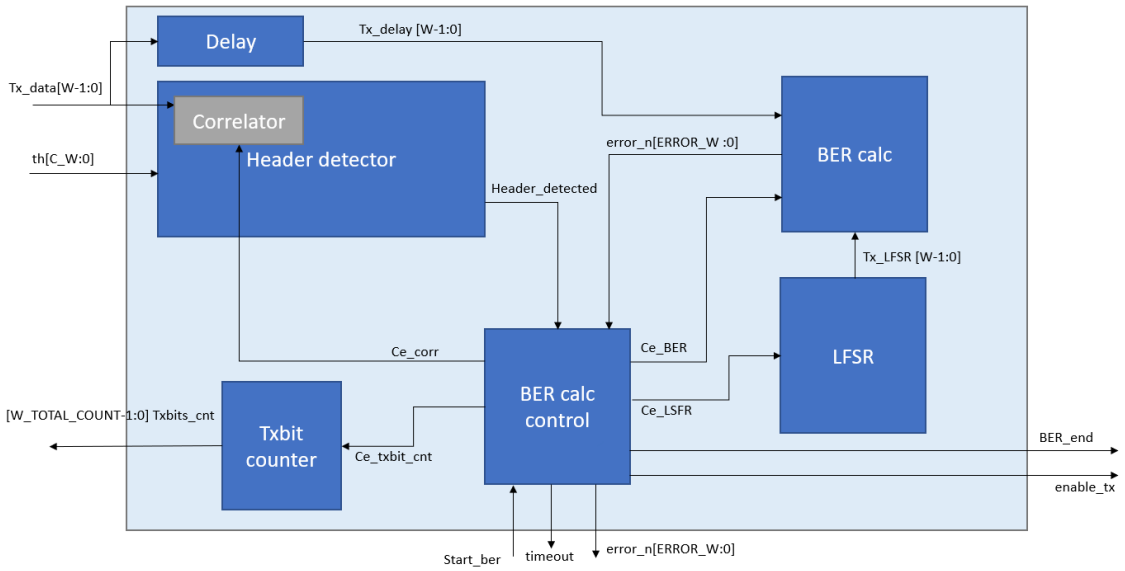


Figura 2.33 Esquema conexiones bloque BER_BLOCK.

El módulo utiliza 5 parámetros de configuración y uno con dependencia a uno de los parámetros.

- W: Bits de la señal de entrada.
- C_W: Ancho de bits de los datos del correlador, se utiliza para la entrada con el valor del umbral.
- W_TOTAL_COUNT: Número de bits del contador de muestras. El tamaño es el de todos los contadores en cascada sumados.
- ERROR_W: Ancho de bits del registro con el número de errores.
- N: Número de bits de la cabecera.
- TX_DELAY: Retardo que se aplica a la señal de entrada, N+1.

Respecto a las entradas y salidas:

- clk: Entrada de reloj.
- start_ber: Entrada que indica el comienzo de la fase de cálculo del BER.
- rst_ber: Reset del sistema de cálculo del BER, activo a nivel alto.
- rst: Reset del sistema, activo a nivel alto.
- Tx_data [W-1:0]: Entrada con los datos recibidos desde el transmisor.
- th [C_W:0]: Entrada con el valor del umbral.
- enable_tx: Señal de habilitación del transmisor.
- timeout: Señal que indica que no se ha detectado la cabecera.
- tx_bits_cnt [W_TOTAL_COUNT-1:0]: Salida con el total de muestras recibidas desde que se recibe la cabecera hasta que se detecta el número de errores determinado.
- timeout: Salida que indica que no se ha detectado la cabecera en el tiempo determinado.
- error_nl[ERROR_W:0]: Número de errores contabilizados.
- ber_end: Señal que indica el fin del proceso de cálculo del BER.

2.3.2.1 Detección de preámbulo

En este módulo HEADER_DETECTOR se detecta el preámbulo que indica el comienzo de la transmisión de datos. Para la detección de la secuencia se utiliza una instancia del correlador diseñado previamente. Se considera que se ha recibido la cabecera cuando el máximo de correlación supera el umbral th calculado en PHASE_RAM. La salida del sistema es una señal binaria que alerta de la recepción de la cabecera cuando está activa.

El módulo utiliza 4 parámetros de configuración:

- C_W: Tamaño de la entrada con el umbral y salida del correlador.
- W: Tamaño de los datos de entrada.
- N: Número de bits de la secuencia con la cabecera.
- C: Configuración del correlador con la secuencia de la cabecera.

Respecto a las entradas y salidas del sistema:

- clk: Entrada de reloj.
- ce_header_detector: Entrada de habilitación del detector de cabecera.
- Tx_data[W-1:0]: Entrada de datos del módulo.
- rst_ber: Reset del sistema de cálculo del BER, activo a nivel alto.
- rst: Reset del sistema, activo a nivel alto.
- th[C_W:0]: Entrada con el umbral que determina si se acepta la detección de la cabecera.
- header_status: Salida del sistema, señal binaria utilizada para indicar la detección de la cabecera.

La parte de la máquina de estados que determina las señales de activación se muestra en la Figura 2.34.

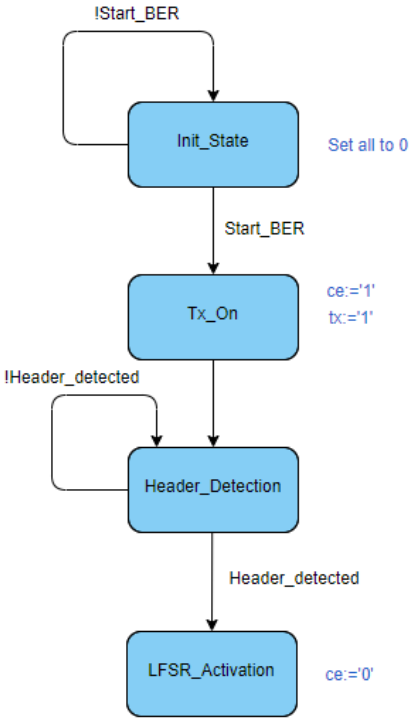


Figura 2.34 Máquina estados HEADER_DETECTOR.

Ce habilita el correlador para que comience a procesar datos de entrada. Al mismo tiempo que ce, se habilita tx para activar el transmisor. Cuando se ha detectado la cabecera, ce se deshabilita.

En la Figura 2.35 se muestra el esquema del módulo.

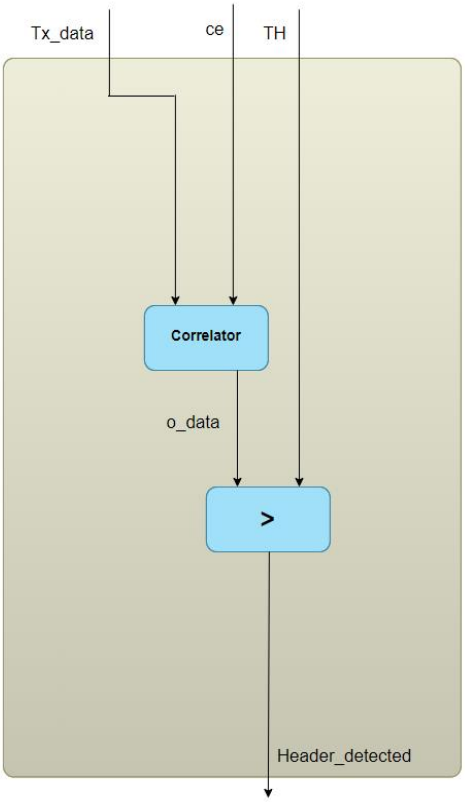


Figura 2.35 Esquema módulo HEADER_DETECTOR.

Se instancia el correlador diseñado previamente. Este recibe los datos de entrada cuando la señal de habilitación está activa. Se compara la salida del correlador con el umbral th. La salida del módulo es header_detected, que está a nivel lógico alto cuando la salida del correlador es mayor al umbral.

2.3.2.2 Contador de errores

El módulo BER_CALC realiza la cuenta de errores de la transmisión de datos recibida. Compara los datos que se han recibido con los que se generan en el LFSR del bloque del cálculo del BER y acumula el número de errores. Los datos recibidos son generados por un LFSR cuyas características son iguales a las del bloque del BER. El ancho de los datos de entrada y del acumulador de errores es parametrizable. Cuenta con un reset propio del bloque del cálculo del BER además del reset global.

El módulo utiliza 2 parámetros de configuración:

- W: Tamaño de los datos de entrada.
- ERROR_W: Tamaño del acumulador de errores.

Respecto a las entradas y salidas del sistema:

- clk: Entrada de reloj.
- ce_ber: Entrada de habilitación del módulo.
- Tx_delay[W-1:0]: Entrada de datos del módulo con la señal recibida.
- Tx_LFSR[W-1:0]: Entrada de datos del módulo con la señal generada por el LFSR.
- rst_ber: Reset del sistema de cálculo del BER, activo a nivel alto.
- rst: Reset del sistema, activo a nivel alto.
- error_n[ERROR_W:0] Salida con el número de errores.

La parte de la máquina de estados que determina las señales de activación se muestra en la Figura 2.36.

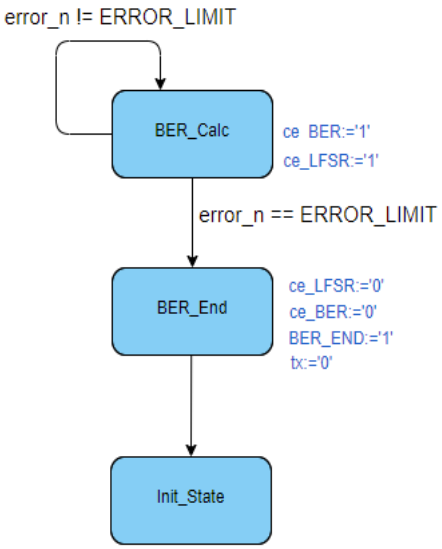


Figura 2.36 Máquina estados BER_CALC.

Ce_BER habilita el módulo BER_CALC y ce_LFSR habilita el LFSR. Cuando el contador de errores alcanza el número configurado en el control, finaliza el cálculo del BER y se deshabilita el LFSR y el módulo BER_CALC.

En la Figura 2.37 se muestra el esquema del módulo.

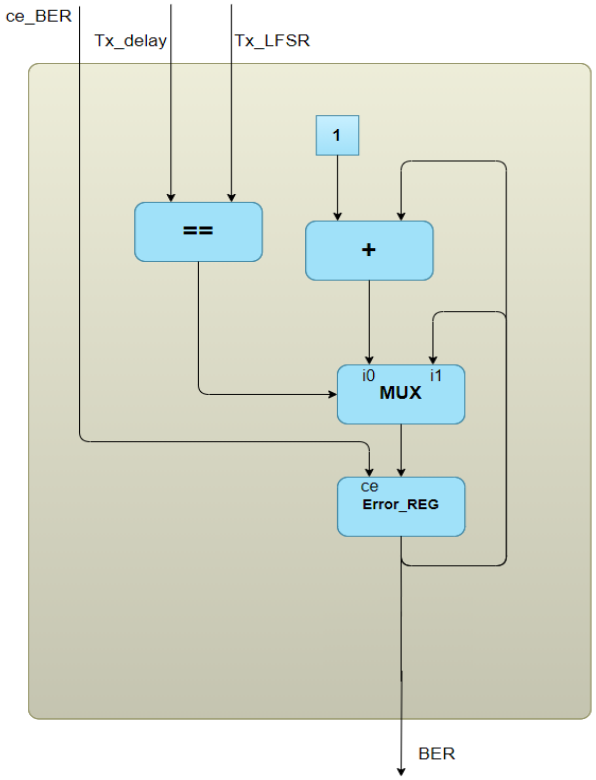


Figura 2.37 Esquema módulo BER_CALC.

Se comparan los dos datos de entrada, cuando son distintos, se incrementa el acumulador de errores.

2.3.2.3 Contador de muestras

El módulo TXBIT_COUNTER, realiza la cuenta de muestras recibidas. El número de muestras transmitidas es de un orden elevado. Así que para mejorar la respuesta temporal del sistema se realiza una configuración de varios contadores en cascada. El número de contadores que contiene el módulo, así como el ancho de bits de los mismos, es configurable.

El módulo COUNTER_CELL, implementa el contador que se instancia repetidas veces en TXBIT_COUNTER. Tiene un parámetro de configuración.

- W_COUNTER_CELL: Tamaño del contador.

Respecto a las entradas y salidas:

- clk: Entrada de reloj.
- ce_counter_cell: Entrada de habilitación del contador.
- rst_ber: Reset del sistema de cálculo del BER, activo a nivel alto.
- rst: Reset del sistema, activo a nivel alto.
- cnt_data [W_COUNTER_CELL-1:0] Salida con la cuenta de muestras.
- overflow: Señal binaria que indica cuando el contador alcanza la cuenta máxima.

En la Figura 2.38 se muestra el esquema de COUNTER_CELL.

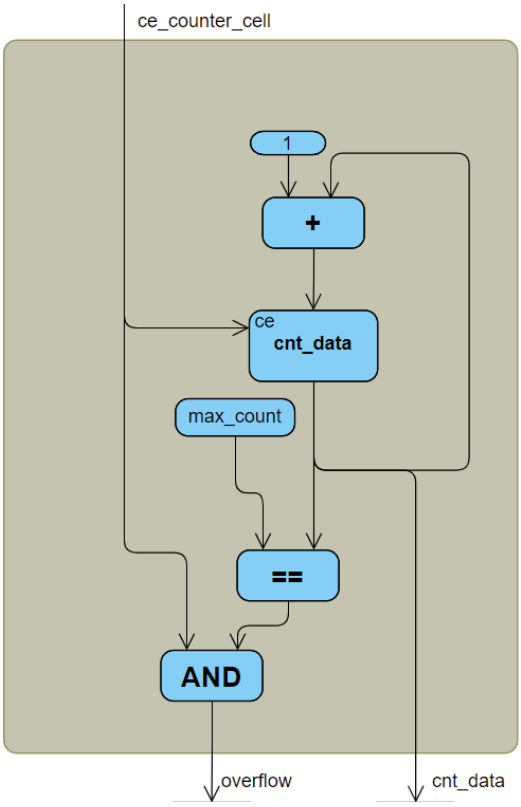


Figura 2.38 Esquema módulo COUNTER_CELL.

Mientras la señal de habilitación está activa, el módulo incrementa la cuenta. Cuando llega al máximo de cuenta se activa la señal de overflow. Esta señal se utiliza para habilitar el siguiente contador. Max_count es una constante que se calcula a partir del parámetro W_COUNTER_CELL, indica el máximo número de cuentas del contador. Max_count = (2^W_COUNTER_CELL)-1.

En el módulo TXBIT_COUNTER, se instancian los contadores COUNTER_CELL en cascada y se combinan los datos de los contadores para obtener el número de cuentas total. Tiene 2 parámetros de configuración y uno que depende de ellos.

- W_COUNTER_CELL: Tamaño del contador.
- COUNTER_N: número de contadores.
- W_TOTAL_COUNT: Ancho de bits de la salida con los datos de todos los contadores se calcula multiplicando los dos parámetros anteriores. W_TOTAL_COUNT = W_COUNTER_CELL * COUNTER_N.

Respecto a las entradas y salidas:

- clk: Entrada de reloj.
- ce_txbits_cnt: Entrada de habilitación del módulo.
- rst_ber: Reset del sistema de cálculo del BER, activo a nivel alto.
- rst: Reset del sistema, activo a nivel alto.
- Txbits_cnt [W_TOTAL_COUNT-1:0] Salida con la cuenta de muestras total.

El esquema de TXBIT_COUNTER puede observarse en la Figura 2.39.

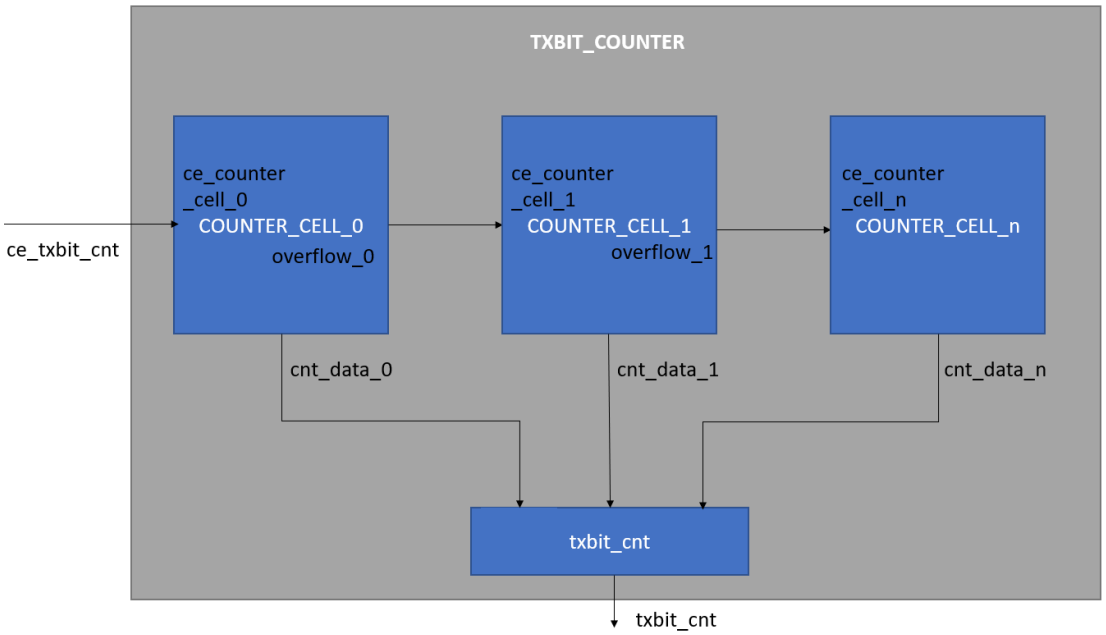


Figura 2.39 Esquema TXBIT_COUNTER para n contadores.

Se instancia el contador múltiples veces con generate. El primer contador comparte la entrada de habilitación con TXBIT_COUNTER, el siguiente, se habilita con la señal de overflow del primer contador y así sucesivamente. Se recoge la información cnt_data de todos los contadores en el registro txbit_cnt. El contador 0 se asigna a los LSB de txbit_cnt y se concatenan los siguientes en orden ascendente.

Para comprobar como la configuración de los contadores afecta a la frecuencia máxima, se sintetiza un contador de 52 bits y TXBIT_COUNTER con diferentes configuraciones cercanas a los 52 bits:

Configuración de los contadores	**Frecuencia máxima**
1*52 (un único contador)	Slow Model Fmax Summary Fmax: 179.18 MHz Restricted Fmax: 179.18 MHz

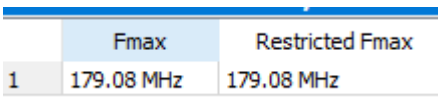
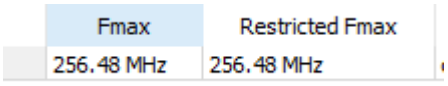
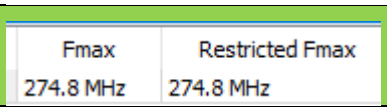
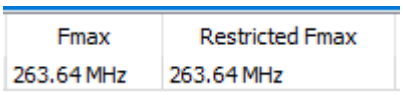
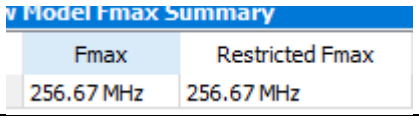
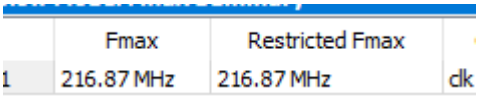
1*52 (TXBIT_COUNTER)	
2*26	
3*18	
4*13	
5*11	
6*9	
8*7	
11*5	
13*4	

Tabla 2.9 Frecuencia máxima del contador para diferentes configuraciones.

La configuración que alcanza la mayor frecuencia de funcionamiento para 52 bits es la de 4 contadores de 13 bits.

2.3.2.4 Control

El control se implementa como una máquina de estados de Moore en el que cada estado proporciona las señales de control de los distintos módulos del cálculo del BER. Además, envía la señal de inicio y detención del transmisor y cuenta con una señal de salida para indicar el final del proceso de cálculo del BER.

La etapa inicial Init_State, pone todos los contadores y señales a 0. Cuando llega la señal de inicio del cálculo del BER, pasa a la siguiente etapa Start_Tx, donde se activa el transmisor y el módulo HEADER_DETECTOR.

Tras un ciclo de reloj, se avanza a la etapa Header_Detection. Donde se incrementa el contador timer_cnt en cada ciclo. El sistema permanece en esta etapa hasta que se recibe la señal Header_Status indicando que se ha detectado la cabecera, o hasta que el contador alcanza el número TIMEOUT determinado mediante un parámetro de configuración.

Si se alcanza TIMEOUT antes de detectar la cabecera, se pasa a la etapa Header_Missed, donde se resetea el contador y se envía la señal timeout al transmisor para que se reinicie la transmisión. La máquina de estados vuelve a la fase inicial.

Si se detecta la cabecera antes del timeout, la etapa que sigue es BER_Calculation. Aquí se deshabilita el módulo HEADER_DETECTOR y se habilita el LFSR, el módulo BER_CALC y TXBIT_COUNTER. El sistema permanece en esta etapa hasta que se alcanza el número de errores configurado en ERROR_LIMIT.

Finalmente, tras alcanzar el número de errores, la última etapa es BER_End. En esta etapa se deshabilitan todos los módulos del sistema y se activa la señal ber_end que indica el final del proceso. El número de errores se fija al parámetro ERROR_LIMIT. De esta forma, si se diera el caso en que ocurre un error en la muestra contigua a la que alcanza ERROR_LIMIT, no se incrementa el número de errores. Tras un ciclo de reloj el control vuelve a la etapa inicial.

En la Figura 2.40 se muestran las etapas del sistema y las condiciones para pasar de una a otra. Las señales y contadores mantienen los valores que tienen en la etapa anterior si no se indica lo contrario.

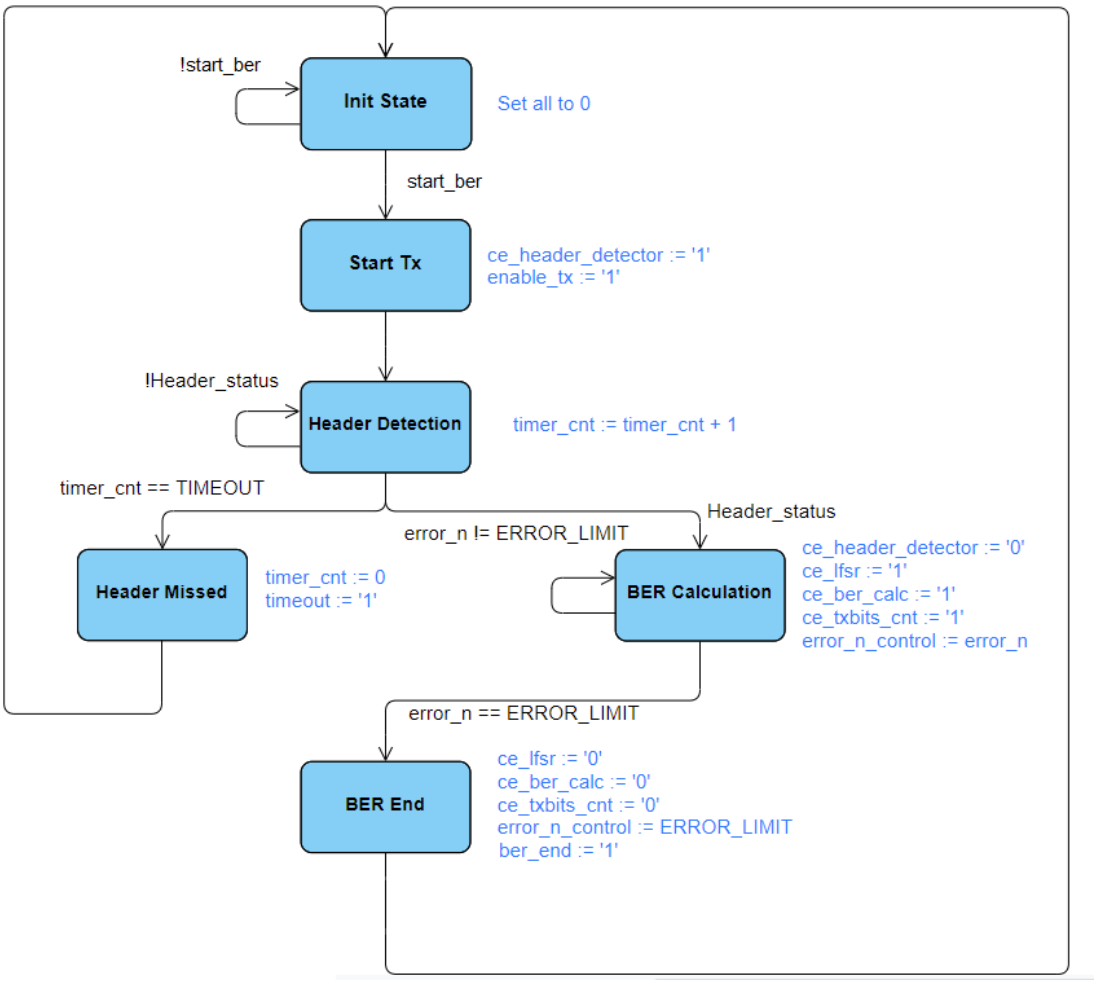


Figura 2.40 Etapas del BER_CONTROL.

El módulo utiliza 2 parámetros de configuración y otros 2 que dependen de estos:

- ERROR_LIMIT: Número de errores a alcanzar para finalizar el cálculo del BER.
- ERROR_W: Número de bits de los registros que cuentan los errores. Es el logaritmo en base 2 de ERROR_LIMIT.
- HEADER_TIMEOUT: Número de muestras en que se considera que se ha perdido la cabecera.
- HEADER_TIMEOUT_W: número de bits del contador timeout. Es el logaritmo en base 2 de HEADER_TIMEOUT.

Respecto a las entradas y salidas:

- clk: Entrada de reloj.
- start_ber: Entrada que indica el comienzo de la fase de cálculo del BER.
- header_status: Entrada que indica si se ha detectado la cabecera.
- error_n[ERROR_W:0] Entrada con el número de errores.
- rst_ber: Reset del sistema de cálculo del BER, activo a nivel alto.
- rst: Reset del sistema, activo a nivel alto.
- enable_tx: Salida de control del transmisor.
- ce_header_detector: Salida de habilitación del módulo HEADER_DETECTOR.
- ce_lfsr: Salida de habilitación del LFSR.
- ce_ber_calc: Salida de habilitación del módulo BER_CALC.
- ce_txbits_cnt: Salida de habilitación del módulo TXBITS_COUNTER.
- ber_end: Salida que indica el fin del proceso de cálculo del BER.
- timeout: Salida que indica que no se ha detectado la cabecera en el tiempo determinado.
- error_n_control[ERROR_W:0]: Número de errores tras procesarlos en el control.

En Tabla 2.10, se muestran las señales que se activan en cada etapa. Si no se especifica en la etapa, la señal mantiene el valor de su etapa anterior.

<table>
<tr><td colspan="2">Init_State</td><td colspan="2">Todo a 0</td></tr>
<tr><td colspan="2">Start_Tx</td><td colspan="2">ce_header_detector = 1
enable_tx = 1</td></tr>
<tr><td colspan="2">Header_Detection</td><td colspan="2">Header_Timeout = Header_Timeout + 1</td></tr>
<tr><td>Header_Missed</td><td>ce_header_detector = 0
enable_tx = 0
Header_Timeout = 0
Timeout = 1</td><td>BER_Calculation</td><td>ce_header_detector = 0
Header_Timeout = 0
ce_lfsr = 1
ce_ber_calc = 1
ce_txbits_cnt = 1
error_n_control = error_n</td></tr>
<tr><td colspan="2">BER_End</td><td colspan="2">ce_header_detector = 0
enable_tx = 0
ce_lfsr = 0
ce_ber_calc = 0
ce_txbits_cnt = 0
ber_end = 1
error_n_control = ERROR_LIMIT</td></tr>
</table>

Tabla 2.10 Activación de señales en cada etapa del BER_CONTROL.

2.3.2.5 Generación de transmisión de referencia

La generación de la señal de referencia se realiza con un LFSR. La configuración del LFSR y la semilla inicial son parametrizables. El bloque cuenta con una señal de entrada para habilitar el sistema y una de salida con la secuencia generada.

El LFSR genera una secuencia pseudo aleatoria que depende de su estructura y su estado inicial (semilla). Se trata de un registro de desplazamiento cuyo bit de entrada es la salida de una función lineal de su estado previo. En el LFSR, el bit de entrada está determinado por puertas XOR entre ciertos puntos del registro y la salida. Estas posiciones del registro son denominadas taps, y para determinar la configuración del LFSR, se indican las posiciones de los taps. La disposición se representa con un polinomio mónico, cuyo módulo 2 es el bit de entrada. El coeficiente con la potencia más elevada indica el número de registros del LFSR, el resto, la posición de los taps [9].

$$x^n + x^{t1} + x^{t2} \ldots + 1$$

Ecuación 2.3 Polinomio LFSR.

Por ejemplo, para $x^5+ x^4+ x^3+1$, la disposición sería mostrada en la Figura 2.41.

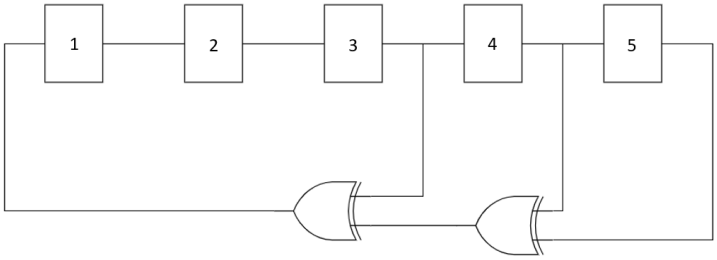


Figura 2.41 Ejemplo LFSR [5,4,3] Fibonacci.

Esta disposición de las puertas XOR se define como Fibonacci LFSR. Puede convertirse al equivalente de Galois ubicando las puertas XOR entre el tap y el siguiente registro. La disposición de Galois es preferible ya que reduce el path al no necesitar puertas XOR concatenadas. Para un LFSR de longitud máxima, la secuencia en las dos formas es la misma, pero desfasada [10].

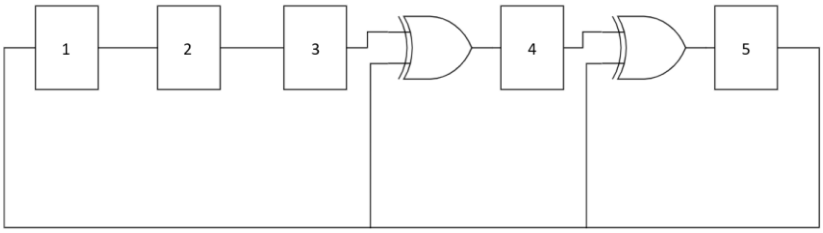


Figura 2.42 Ejemplo LFSR [5,4,3] Galois.

El LFSR genera una secuencia de longitud máxima (2^n-1) cuando el polinomio es primitivo. Para que esto ocurra debe cumplirse que el número de taps sea par y que los taps sean coprimos. Por ejemplo, el LFSR x^6+x^5+1 tendría una longitud máxima, obteniendo a la salida un periodo de 63 bits. La secuencia generada varía en función de la semilla.

En este módulo, se ha utilizado la configuración de Galois. Se crea un registro del mismo tamaño que el orden del polinomio que define el LSR y se realiza una operación XOR en los taps.

El módulo utiliza 3 parámetros de configuración:

- O: Tamaño del registro LFSR, coincide con el polinomio de máximo exponente.
- P: Posición de los taps. Los unos indican que hay un tap en esa posición. Por ejemplo 11000 corresponde a x^4+x^3+1. El LSB representa al 1 y no se tiene en cuenta, se utiliza para que x^1 esté en la posición 1 del parámetro. En Matlab se genera un fichero con la codificación del polinomio. El MSB no se utiliza, como se ve en la Figura 2.42, en la última posición no se genera un tap, está puesto por identificar el polinomio con más claridad.
- I: Semilla inicial.

Respecto a las entradas y salidas del sistema:

- clk: Entrada de reloj.
- ce: Entrada de habilitación del módulo.
- rst: Reset del sistema, activo a nivel alto.
- R_seq: Salida del sistema, dato de la última posición del registro LFSR.

Se crea un registro Q de tamaño O. Cuando se reinicia el módulo, se inicializa con la semilla almacenada en I. Cuando el módulo está habilitado, se comporta como un registro de desplazamiento circular, solo que, en la posición de los taps, se realiza una operación XOR entre el dato del registro anterior y la salida. Se recorre el parámetro P para localizar los taps.

Un punto a tener en cuenta al testear el LFSR, es que PNSequence de Matlab genera una secuencia con el modelo de Fibonacci por lo que la secuencia en Matlab y Hardware no coinciden salvo para un caso. Empíricamente se observó que para la semilla [1 0 … 0] El modelo de Galois y el de Fibonacci coinciden. Así que, para poder realizar pruebas con otras semillas con Galois, se ha de modelar el LFSR en Simulink.

2.3.2.6 Verificación del bloque del BER

Desde Matlab se obtienen ficheros de texto con los datos utilizados para la simulación.

- Se utiliza el fichero con los datos de la transmisión completa: preámbulo y transmisión válida tras el filtrado y el ruido.
- Fichero con el valor del umbral.
- Fichero con el número de errores.

Para el test se utiliza un paquete generado en Matlab con los parámetros de configuración. En ModelSim se genera un banco de pruebas para testear el sistema completo.

En el bloque initial se abren los ficheros de texto, se inicializan las señales y se carga el umbral. A continuación, se resetea el sistema y se activa la señal de inicio del sistema del cálculo del BER.

Cuando se recibe la señal de habilitación del transmisor, se comienza la lectura de datos de entrada desde el fichero. Si se llega al final del fichero, se escribe un mensaje indicándolo y mostrando el número de errores entre la señal recibida y la generada por el LFSR.

En caso de recibirse la señal de timeout, se indica en pantalla que no se ha recibido la cabecera.

Cuando se recibe la señal de fin del proceso de cálculo del BER, se compara que el número de errores en bloque hardware coinciden con los obtenidos en Matlab. Se resta uno al número de bits obtenidos en el hardware porque el sistema se incrementa también una muestra después de alcanzar el número de errores determinado. Tras realizar la simulación, se muestra en pantalla si los resultados de ambos modelos coinciden.

Se realiza una serie de test para comprobar que el sistema funciona bajo diferentes condiciones. Las pruebas se realizan con un ruido elevado para llegar al número de errores configurado.

Variaciones en el número de errores hasta detener el cálculo del BER

Se modifica el número de errores que el sistema detecta hasta finalizar el proceso del cálculo del BER. Se aprovecha este apartado para realizar pruebas con LFSR de distintos ordenes ya que se necesita un mayor número de muestras al aumentar el límite de errores.

- Configuración de los contadores en cascada: 2 contadores de 8 bits.

Número de errores	**Orden LFSR**	**Resultado**
1	7	`# Number of errors 1, of 28 transmitted bits`
50	11	`# Number of errors 50, of 1158 transmitted bits`
200	13	`# Number of errors 200, of 5155 transmitted bits`
1000	15	`# Number of errors 1000, of 25443 transmitted bits`

Tabla 2.11 Test variando el número de errores hasta detener el sistema y el orden del LFSR.

Variaciones en la configuración de los contadores

Se realizan las pruebas modificando el número de contadores en cascada y el número de bits de estos.

- Número de errores hasta detener el cálculo del BER: 40

Configuración contadores	**Resultado**
1 contador de 16 bits	# Number of errors 40, of 967 transmitted bits
4 contadores de 4 bits	# Number of errors 40, of 956 transmitted bits
4 contadores de 13 bits	# Number of errors 40, of 831 transmitted bits
9 contadores de 6 bits	# Number of errors 40, of 1006 transmitted bits
16 contadores de 1 bit	# Number of errors 40, of 826 transmitted bits

Tabla 2.12 Test variando la configuración de los contadores en cascada.

Test modificando la cabecera

Se realiza una serie de pruebas modificando la cabecera. En el primer test se modifica la cabecera transmitida para que no coincida con la configurada en el correlador y así poder comprobar que se activa la señal de timeout.

- Número de errores hasta detener el cálculo del BER: 40
- Configuración de los contadores en cascada: 2 contadores de 8 bits.

Polinomio cabecera	**Resultado**
Cabecera transmitida distinta	# header sequence not detected
5,3,0	# Number of errors 40, of 1033 transmitted bits
6,5,0	# Number of errors 40, of 1089 transmitted bits
9,5,0	# Number of errors 40, of 785 transmitted bits

Tabla 2.13 Test variando la cabecera.

Test modificando el ruido

Se realizan pruebas modificando el factor de ruido para comprobar como varía el BER. En el último caso no se llega al número de errores definido antes de alcanzar el final del fichero.

Número de errores hasta detener el cálculo del BER: 40

Configuración de los contadores en cascada: 2 contadores de 8 bits.

SNR_dB	**Resultado**	**BER**

2	Number of errors 40, of 642 transmitted bits	6.23%
4	Number of errors 40, of 1071 transmitted bits	3.73%
6	Reached end of the file, errors = 29	---

Tabla 2.14 Test variando el ruido.

2.3.3 Transmisor

Es el bloque encargado de generar las señales que se transmiten a través del LED de iluminación. Dispone de dos modos de operación, el primero genera las señales utilizadas en la fase de calibración y el segundo las utilizadas en el cálculo del BER. Una señal de control indica al transmisor que tipo de secuencia generar. Se utiliza una señal de habilitación para determinar si el transmisor genera una de las secuencias o no envía señales.

La configuración de las señales generadas se realiza mediante parámetros de configuración. Estas se ajustan a los requerimientos fijados en el diseño de los bloques de calibración y cálculo del BER.

El sistema se subdivide en distintos bloques, que se detallan más adelante, siguiendo la siguiente jerarquía:

- TRANSMISSOR
  - TRANSMISSION_CALIBRATION_MODE
    - TRANSMISSION_CALIBRATION
    - TRANSMISSION_CALIBRATION_CONTROL
  - TX_BER_MODE
    - TX_BER
    - TX_BER_CONTROL

Utiliza 1 parámetro de configuración:

- W: Ancho de la secuencia de salida.

Respecto a las entradas y salidas:

- clk: Entrada de reloj.
- Tx_mode: Entrada con la señal que indica el modo de funcionamiento.
- rst: Reset del sistema, activo a nivel alto.
- rst_ber: Reset de la fase del cálculo del BER.
- enable_tx: Entrada que habilita la transmisión.
- ber_end: Entrada que indica el fin del cálculo del BER.
- Tx_data[W-1:0] Salida con la señal a transmitir.

Si la señal enable_tx está habilitada y Tx_mode es 0, está activo el modo de calibración. Si Tx_mode es 1, el modo del BER. La salida del sistema es la salida del módulo TRANSMISSION_CALIBRATION_MODE cuando Tx_mode es 0. Cuando Tx_mode es 1, es la salida de TX_BER_MODE. En la Figura 2.43 se muestra el esquema del transmisor.

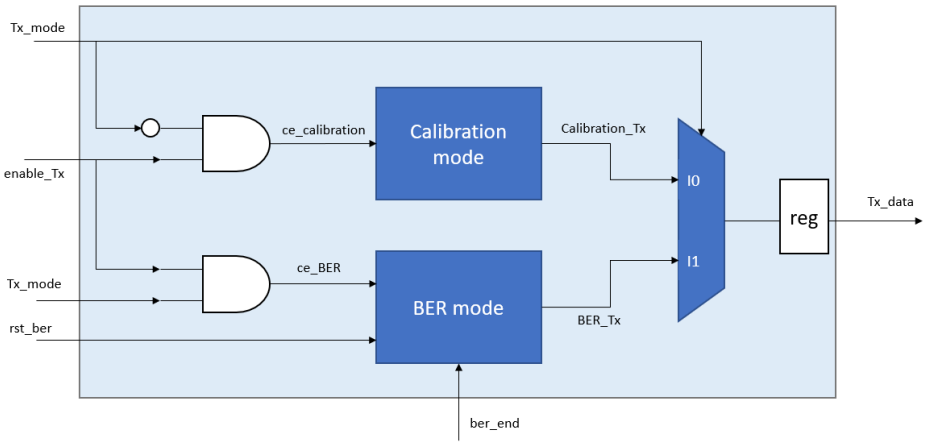


Figura 2.43 Esquema del transmisor.

2.3.3.1 Modo Calibración

Este módulo genera las secuencias utilizadas en la fase de calibración. Esta secuencia consiste en una señal de calibración determinada seguida de un número determinado de ceros. Este patrón se repite constantemente mientras el transmisor esté funcionando en modo calibración. En la capa superior del módulo se realizan las conexiones entre la máquina de estados y el módulo que genera la señal de calibración.

Se utiliza 1 parámetro de configuración.

- W: Bits de la señal de entrada.

Respecto a las entradas y salidas:

- clk: Entrada de reloj.
- ce_calibration: Entrada de habilitación del bloque.
- rst: Reset del sistema, activo a nivel alto.
- calibration_Tx [W-1:0]: Salida con la secuencia generada.

El módulo TRANSMISSION_CALIBRATION es la parte encargada de generar las señales a transmitir en el proceso de calibración. La secuencia de calibración se obtiene mediante el LFSR diseñado previamente.

Cuenta con 4 parámetros de configuración:

- O: Orden del LFSR.
- P: Polinomio de configuración del LFSR.
- I: Semilla inicial del LFSR.
- W: Bits de la señal de entrada.

Respecto a las entradas y salidas:

- clk: Entrada de reloj.
- ce_lfsr: Entrada de habilitación del LFSR.
- zero_tx: Señal que indica si se envía la secuencia de calibración o ceros.
- rst: Reset del sistema, activo a nivel alto.
- calibration_Tx [W-1:0]: Salida con la secuencia de calibración generada.

Se utiliza la señal zero_tx para resetear el LFSR y como selector del multiplexor que determina la salida del bloque. Si ce_lfsr es activa y zero_tx inactiva, se envía la secuencia generada por el LFSR, en el caso que zero_tx sea activa, se envían ceros. En la Figura 2.44 se observa el esquema.

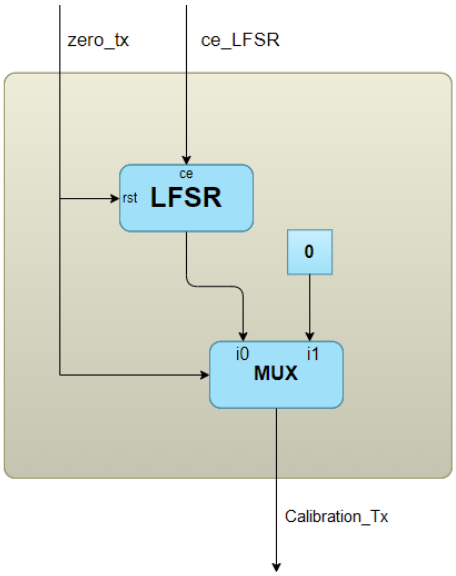


Figura 2.44 Esquema módulo TRANSMISSION_CALIBRATION.

El módulo TRANSMISSION_CALIBRATION_CONTROL genera las señales de control.

Utiliza 3 parámetros de configuración:

- N: Número de bits de la secuencia de calibración
- C_W: Ancho de la secuencia de calibración.
- SC: Numero de muestra a contar por la fase de calibración, incluyendo los ceros.

Respecto a las entradas y salidas:

- clk: Entrada de reloj.
- ce_calibration: Entrada de habilitación que indica el inicio de la fase de calibración.
- rst: Reset del sistema, activo a nivel alto.
- ce_lfsr: Señal que indica que se envía la secuencia de calibración.
- zero_tx: Señal que indica que se envían ceros.

En la Figura 2.45 se muestra el diagrama de la máquina de estados.

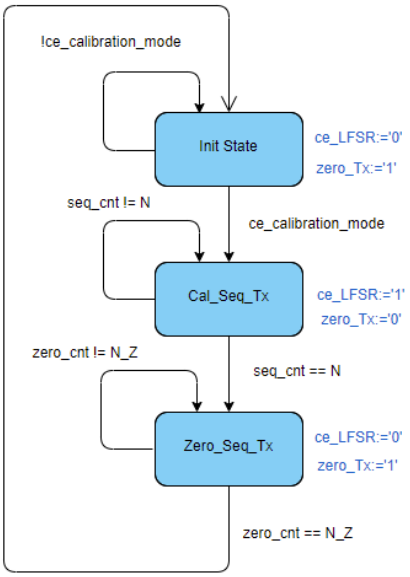


Figura 2.45 Diagrama FSM TRANSMISSION_CALIBRATION_CONTROL.

Cuando se activa la señal ce_calibration, la máquina de estados avanza del estado inicial, en este estado se activa zero_Tx para que se obtengan ceros a la salida. Esto se hace porque cuando el módulo esté funcionando, pasará por este estado durante un ciclo de reloj en cada vuelta. En este estado se envía el cero extra necesario para compensar el momento en el que el sistema de calibración del receptor pasa por un estado en el que no cuenta muestras.

En el estado Cal_Seq_Tx está activa la señal ce_LFSR. El LFSR genera la señal de calibración hasta que se cuenta el número de muestras N. En este momento, se pasa al estado Zero_Seq_Tx donde se habilita zero_Tx para que el sistema envíe ceros durante N_Z ciclos. Siendo N_Z el número de ceros configurado, SC-N.

2.3.3.2 Modo cálculo del BER

Este bloque genera las secuencias utilizadas en la fase del cálculo del BER. Esta secuencia consiste en una señal de cabecera que indica el inicio de la transmisión y, a continuación, una señal generada por un LFSR que está en marcha continua hasta que recibe la señal de fin de cálculo del BER. En la capa superior del módulo se realizan las conexiones entre la máquina de estados y el módulo que genera la señal de la fase del BER.

Se utiliza 1 parámetro de configuración.

- W: Bits de la señal de entrada.

Respecto a las entradas y salidas:

- clk: Entrada de reloj.
- ce_ber: Entrada de habilitación del bloque.
- rst: Reset del sistema, activo a nivel alto.
- rst_ber: Reset de la parte del BER.
- ber_end: Entrada que indica el fin del cálculo del BER.
- ber_Tx [W-1:0]: Salida con la secuencia generada.

TX_BER genera la señal utilizada para realizar el cálculo del BER.

El módulo cuenta con 7 parámetros de configuración:

- O: Orden del LFSR de la cabecera.
- P: Polinomio de configuración del LFSR de la cabecera.
- I: Semilla inicial del LFSR de la cabecera.
- O_BER: Orden del LFSR de la señal transmitida.
- P_BER: Polinomio de configuración del LFSR de la señal transmitida.
- I_BER: Semilla inicial del LFSR de la señal transmitida.
- W: Bits de la señal de entrada.

Respecto a las entradas y salidas:

- clk: Entrada de reloj.
- ce_lfsr_header: Entrada de habilitación del LFSR de la cabecera.
- ce_lfsr_ber: Entrada de habilitación del LFSR de la señal transmitida.
- lfsr_selector: Entrada que indica LFSR se utiliza a la salida.
- Rst: Reset del sistema, activo a nivel alto.
- rst_ber: Reset de la fase del cálculo del BER.
- ber_Tx [W-1:0]: Salida con la secuencia generada.

Se instancian dos LFSR para generar la secuencia. Con la señal lfsr_selector se selecciona el LFSR que se utiliza para la salida del módulo. Si es cero, se utiliza el LFSR con la cabecera, si es uno, el LFSR con la señal a transmitir. En Figura 2.46 muestra el esquema del módulo.

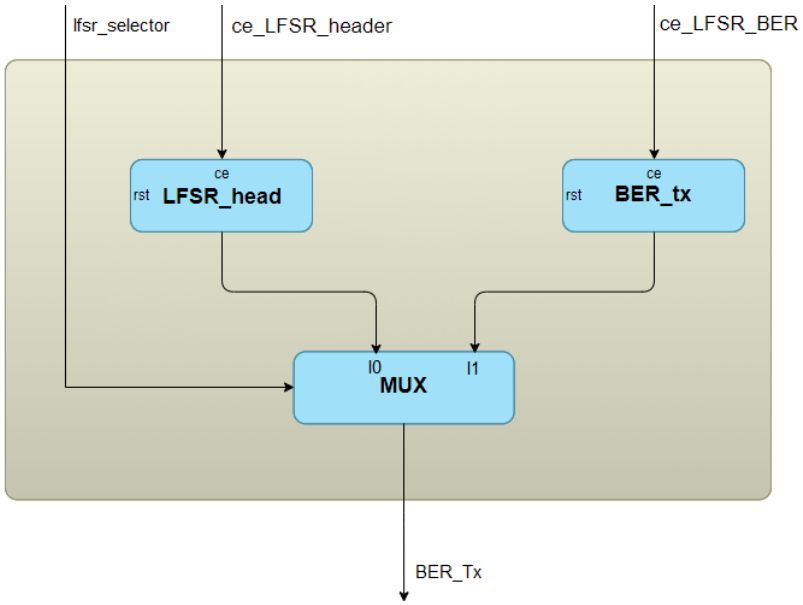


Figura 2.46 Esquema módulo TX_BER.

TX_BER_CONTROL genera las señales de control de TX_BER.

Utiliza 2 parámetros de configuración:

- N: Número de bits de la secuencia de calibración
- C_W: Ancho de la secuencia de calibración.

Respecto a las entradas y salidas:

- clk: Entrada de reloj.
- ce_ber: Entrada de habilitación que indica el inicio de la fase de cálculo del BER.
- rst: reset del sistema, activo a nivel alto.
- rst_ber: Reset de la fase del cálculo del BER.
- ber_end: Entrada que indica el final de la fase del BER.
- ce_lfsr_header: Salida que indica que se envía la secuencia con la cabecera.
- ce_lfsr_ber: Salida que indica que se envía la secuencia sobre la que se realiza el cálculo del BER.
- lfsr_selector: Salida que indica que lfsr se elige como salida del módulo.

En la Figura 2.47 se muestra el diagrama de la máquina de estados.

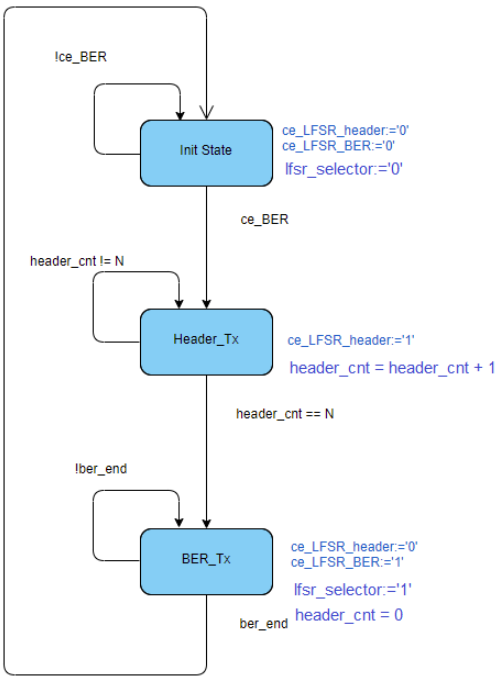


Figura 2.47 Máquina de estados TX_BER_CONTROL.

Cuando se recibe la señal ce_ber, la máquina de estados abandona la etapa Init_State, dónde todas las señales se han inicializado a 0.

En Header_Tx, se activa ce_lfsr_header, que habilita el LFSR con la secuencia de cabecera. Además, se incrementa el contador header_cnt. Cuando este alcanza el valor N, se han generado la cabecera completa y se pasa a la siguiente etapa.

En BER_Tx, se ponen las señales anteriores a 0 y se activan ce_lfsr_ber y lfsr_selector para cambiar el lfsr y la salida del módulo BER_TX. La máquina permanece en este estado hasta que se recibe la señal ber_end, en ese momento regresa a la etapa inicial.

2.3.4 Incremento de fase

Es el bloque encargado de incrementar la fase del reloj que se utiliza para la recepción de datos. Este se utiliza para realizar la captura de la señal recibida en distintas fases. Se genera otro reloj adicional de la misma frecuencia, que se mantiene en fase constante y es utilizado por el resto del sistema.

Una señal de entrada indica que se requiere incrementar la fase del reloj de captura de datos, esta misma señal pone en marcha la máquina de estados que gestiona el incremento de fase del reloj dinámico del PLL. Además, es posible recibir un número de fase y el sistema se desplaza hasta esa fase indicada. Esto se realiza una vez se ha determinado cuál es la fase óptima en el bloque de calibración.

Para realizar el incremento de fase, se utiliza el PLL Intel FPGA del catálogo de IP de Quartus Prime. Se generan 4 pares de relojes para poder seleccionar 4 frecuencias distintas.

El módulo superior del sistema es INCREMENT_PHASE, este recibe las señales de incremento de fase y selección de fase. Transfiere las señales de control a la máquina de estados que se

encarga del cambio dinámico de fase, y controla el proceso de desplazamiento del reloj hasta la fase óptima. Este bloque funciona bajo el mismo reloj de referencia que el PLL, mientras que el resto del sistema utilizan los relojes generados por el PLL. Por esta razón las señales que cruzan los dominios de reloj se tratan de forma especial para asegurar que el sistema no pierde sincronía.

Se utilizan seis parámetros de configuración.

- PH: Número de fases.
- P_W: Ancho de bits del número de fases.
- N_STEP0, N_STEP1, N_STEP2, N_STEP3: Número de incrementos necesarios para incrementar la fase una unidad, un parámetro para cada reloj frecuencia posible.

Respecto a las entradas y salidas:

- refclk: Entrada de reloj.
- scanclk: Free running clock.
- rst: Reset del sistema, activo a nivel alto.
- inc_ph: Entrada para incrementar la fase. Esta señal cruza dominios de reloj, desde el bloque que la genera se ha mantenido activa durante varios ciclos de reloj.
- pll_select: Entrada que indica que relojes del PLL se utilizan.
- optimal_ph [P_W:0] Entrada con la fase óptima.
- set_ph: Entrada para indicar que se desfase hasta la fase óptima.
- inc_ph_done: Salida que indica que se ha incrementado la fase. Cruza dominios de reloj, por lo que se mantiene activa durante varios ciclos de reloj.
- set_ph_done: Salida que indica que el reloj se ha desfasado hasta la fase óptima.

En la Figura 2.48 se muestra el esquema del bloque.

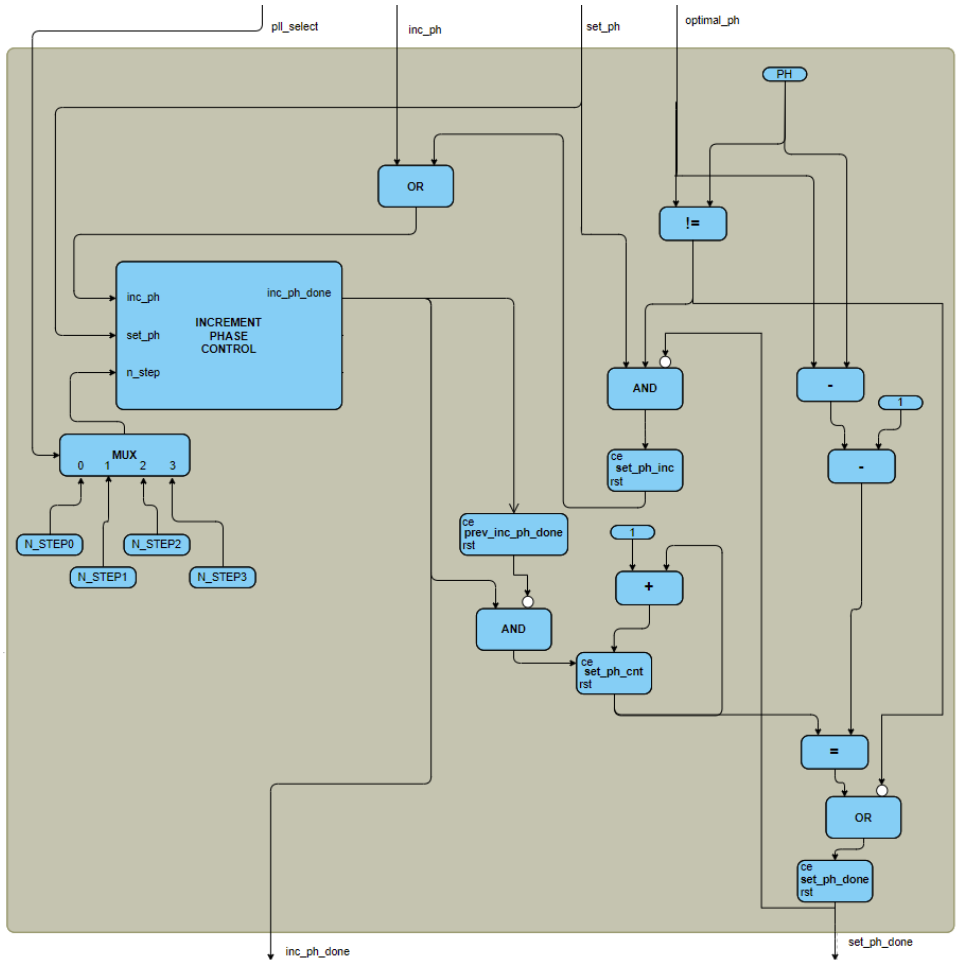


Figura 2.48 Esquema bloque INCREMENT_PHASE.

INCREMENT_PHASE_CONTROL es el módulo que se encarga de incrementar la fase del reloj outclk1, este módulo se detalla más adelante.

La señal set_ph indica que se desfase el reloj outclk1 hasta la fase óptima. Cuando la señal set_ph está activa, la fase óptima es distinta de la última fase y el registro set_ph_done es 0, el registro set_ph_inc se activa.

Esta señal está conectada a INCREMENT_PHASE_CONTROL, que realiza un incremento de fase cuando set_ph_inc o inc_ph es 1. El módulo activa la señal inc_ph_done durante varios ciclos de reloj una vez finalizado el incremento de fase, pero se necesita capturar el flanco de subida únicamente. Para ello se utiliza el registro prev_inc_ph_done, cuyo valor es el de la salida de inc_ph_done. Con una puerta and de la inversa de prev_inc_ph_done y inc_ph_done, se captura el flanco de subida de inc_ph_done. Cuando esto ocurre, se incrementa el contador set_ph_cnt.

Se repite este proceso hasta que el contador sea igual a la diferencia entre el número de fases y la fase óptima menos 1. Esto es así porque cuando finaliza el proceso de calibración, el desfase de outclk1 es igual al número de fases. Sabiendo que está en la última fase, se decrementa de fase en orden descendente hasta alcanzar la fase óptima. El -1 es porque el contador se incrementa una vez finalizado el decremento de fase, por lo que hay que realizar el proceso una vez menos de lo que indica el contador.

Cuando se llega a la fase óptima, se activa la señal set_ph_done para indicar que se ha finalizado este proceso. Esto también ocurre cuando la fase óptima coincide con la última fase.

La salida inc_ph_done se obtiene del módulo de control.

2.3.4.1 PLL

En este módulo, se instancia el PLL de fase dinámica generado con la IP de Altera. Este permite ajustar la fase de los relojes generados en función del reloj de referencia y las mismas salidas de reloj del PLL. El mínimo incremento de fase es 1/8 del periodo VCO [11].

Se utiliza un parámetro de configuración.

- PLL_W: Ancho del dato utilizado para seleccionar el par de relojes del PLL.

Respecto a las entradas y salidas:

- refclk: Entrada de reloj.
- scanclk: Free running clock.
- rst_pll: Reset del pll.
- phase_en: Entrada para solicitar el incremento de fase mínimo.
- set_ph: Entrada para indicar que se desfase hasta la fase óptima. En este caso se conecta la inversa de esta señal a la entrada updn del PLL para que realice el desfase en la otra dirección.
- pll_select: Entrada que indica que relojes del PLL se utilizan.
- set_ph_done: Salida que indica que el reloj se ha desfasado hasta la fase óptima.
- outclk_0: Reloj fijo utilizado por el transmisor.
- outclk_1: Reloj de fase dinámica utilizado por el receptor.
- phase_done: Salida que indica que se ha realizado el incremento de fase mínimo.

En la Figura 2.49 se muestra el esquema.

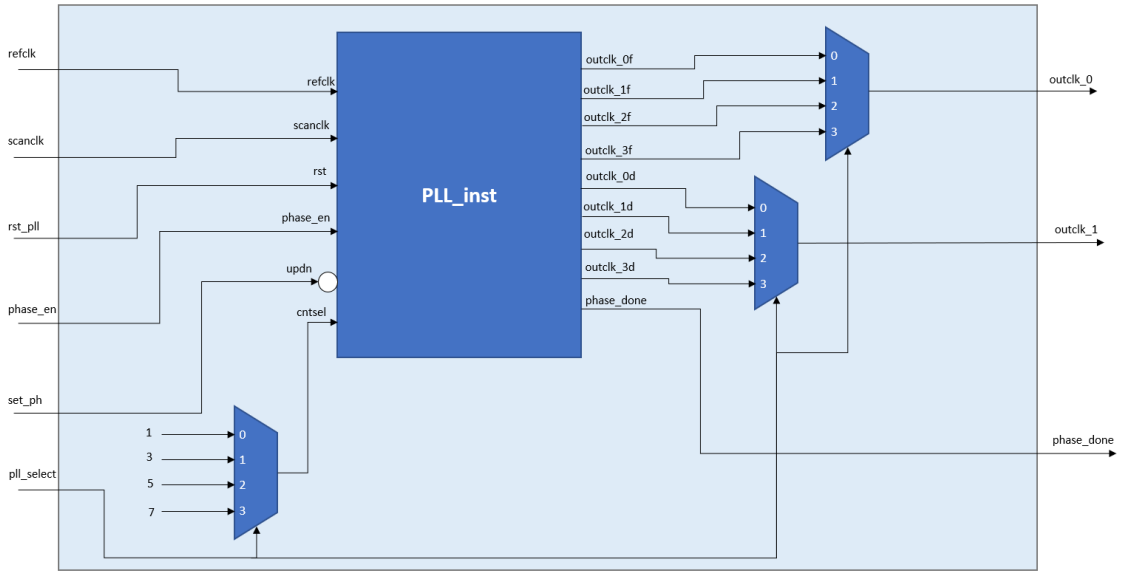


Figura 2.49 Esquema de PLL.

En base a pll_select, se selecciona qué reloj se desfasa configurando la señal cntsel del PLL, y el par de relojes que se seleccionan como salida del módulo. Se generan 4 pares de relojes con el PLL, de estos uno corresponde al reloj estático, y el otro al reloj dinámico.

2.3.4.2 Control de incremento de pasos del PLL

Este módulo se encarga de incrementar la fase del reloj outclk_1. Se realiza siguiendo la nota de aplicación an661 de Altera en referencia a los PLL de fase dinámica como se muestra en la Figura 2.50.

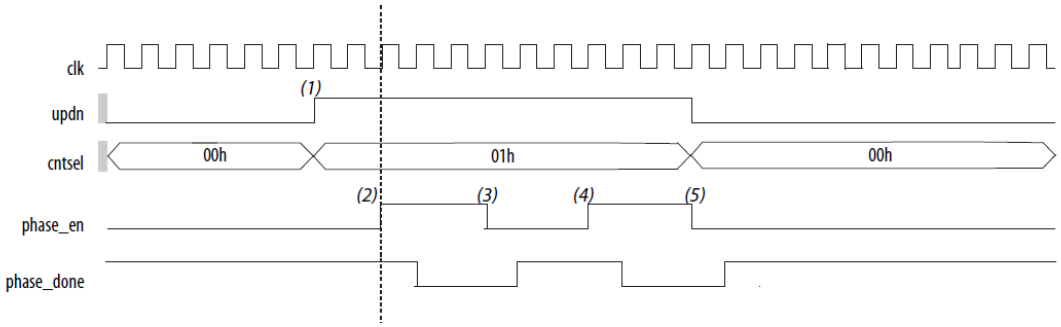


Figura 2.50 configuración señales PLL fase dinámica. Fuente [11].

El PLL se incrementa en pasos de 1/8 del periodo del VCO configurado. Para realizar este incremento, primero se elige el reloj que se va a incrementar con el registro cntsel, para el reloj 1 es el 5'b1. Updn se pone a 1 para incrementar la fase, en caso contrario decrementa. Se habilita la señal phase_en durante al menos dos ciclos de reloj. Finalmente, se deshabilita una vez phase_done es cero. Para incrementar un paso más, se vuelve a habilitar phase_en.

Como los incrementos son de 1/8 de VCO, cada cambio de fase corresponde a un número determinado de estos pequeños pasos, en función del número de fases que se desee configurar. Conociendo que el paso mínimo es Tstep_min = 1/(Fvco*8), el número máximo de pasos en un periodo del reloj outclk_1 de frecuencia Fo es Nstep_max = 1/(Tstep_min*Fo). Este número se divide por el número de fases para saber cuántas activaciones de phase_en son necesarias para incrementar una fase. Nstep_ph = Nstep_max/PH.

Esta operación de incremento de fase es la que se realiza en el módulo INCREMENT_PHASE_CONTROL, mediante una máquina de estados como se muestra en la Figura 2.51.

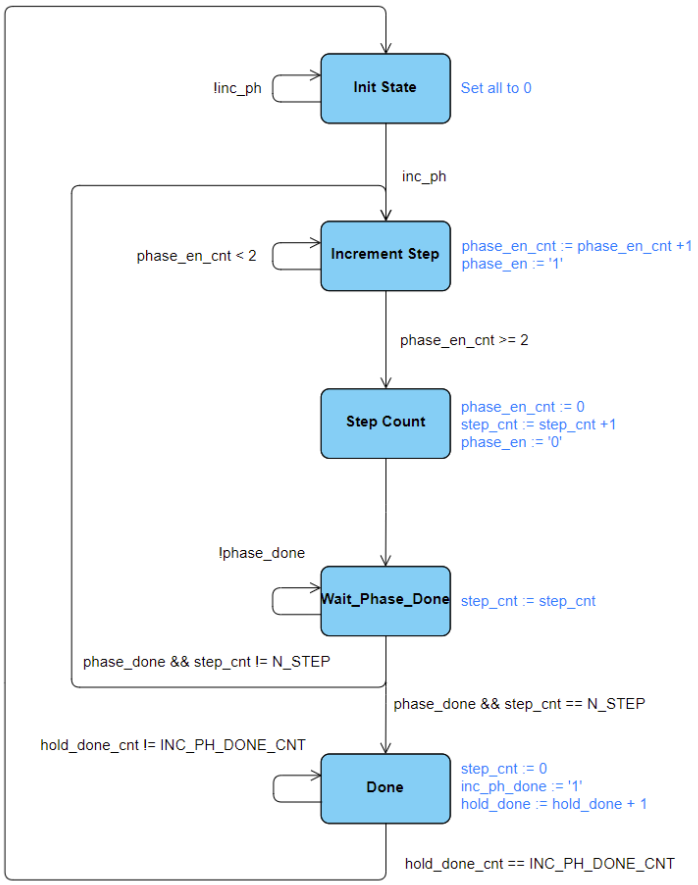


Figura 2.51 Máquina de estados INCREMENT_PHASE_CONTROL.

En el estado Init_State se reinician todos los registros y señales. Cuando se recibe la señal de incremento de fase inc_ph, se pasa a la etapa Increment_Step.

En esta etapa se habilita phase_en del PLL para incrementar la fase 1/8 del periodo de VCO. Además, se incrementa el contador que se asegura que la señal está habilitada durante dos ciclos de reloj tal y como se indica en la nota de aplicación de Altera.

Una vez se ha mantenido phase_en activa durante 2 ciclos, se deshabilita y se reinicia el contador phase_en_cnt, a su vez se incrementa el contador de pasos. Tras un ciclo de reloj se pasa a la etapa Wait_Phase_Done.

Si la señal phase_done no se ha activado, se espera en esta etapa. En caso contrario, si el contador de pasos alcanza el número de pasos necesarios para el incremento de fase, definido por N_STEP, se pasa a la etapa Done. Mientras no se alcance ese número se vuelve a la etapa Increment_Step para repetir la habilitación de phase_en.

En última etapa Done, se reinicia el contador de pasos step_cnt y se activa la señal inc_ph_done para indicar que se ha finalizado el incremento de fase se mantiene la señal activa durante INC_PH_DONE_CNT. Esta señal se genera con el reloj de 50MHz, pero debe ser captada por el generado por el PLL. Por ello se mantiene activa durante un número de ciclos determinado por el parámetro. Este se calcula como la división entre el reloj de 50 MHz y el reloj más lento generado, el resultado se multiplica por 2. Finalmente, la máquina de estados vuelve a la etapa inicial tras alcanzar la cuenta INC_PH_DONE_CNT.

El módulo utiliza dos parámetros de configuración:

- P_W: Ancho de bits del número de fases.
- INC_PH_DONE_CNT: Número de muestras que la señal inc_ph_done debe permanecer activa.

Respecto a las entradas y salidas:

- clk: Entrada de reloj.
- rst: Reset del sistema, activo a nivel alto.
- phase_done: Entrada que indica que se ha realizado el incremento de fase mínimo.
- n_step: Entrada con el número de pasos necesarios para cambiar de fase.
- inc_ph: Entrada para incrementar la fase.
- phase_en: Salida para indicar el incremento de fase mínimo.
- inc_ph_done: Salida que indica que se ha incrementado la fase.

2.3.5 Captura y procesado de la señal de entrada

El sistema funciona bajo el reloj fijo generado por el PLL, pero la lectura de datos desde el LED se realiza mediante el otro reloj con desfase dinámico generado por el PLL. Para cruzar los datos entre los dos dominios, se utiliza una memoria first in, first out (FIFO) asíncrona. La escritura de datos en la FIFO se realiza bajo el reloj dinámico. La lectura se realiza bajo el reloj base y concurrentemente a la escritura.

Se utiliza un parámetro de configuración.

- W: Ancho de bits de los datos.

Respecto a las entradas y salidas:

- data_in [W-1:0]: Entrada de datos.
- enable_tx: Señal de entrada utilizada para solicitar la escritura en la FIFO.
- rdclk: Reloj de lectura, ligado al reloj estático del PLL.
- wrclk: Reloj de escritura, ligado al reloj dinámico del PLL.
- data_out [W-1:0]: Salida de datos.

En la Figura 2.52 se muestra el esquema del módulo.

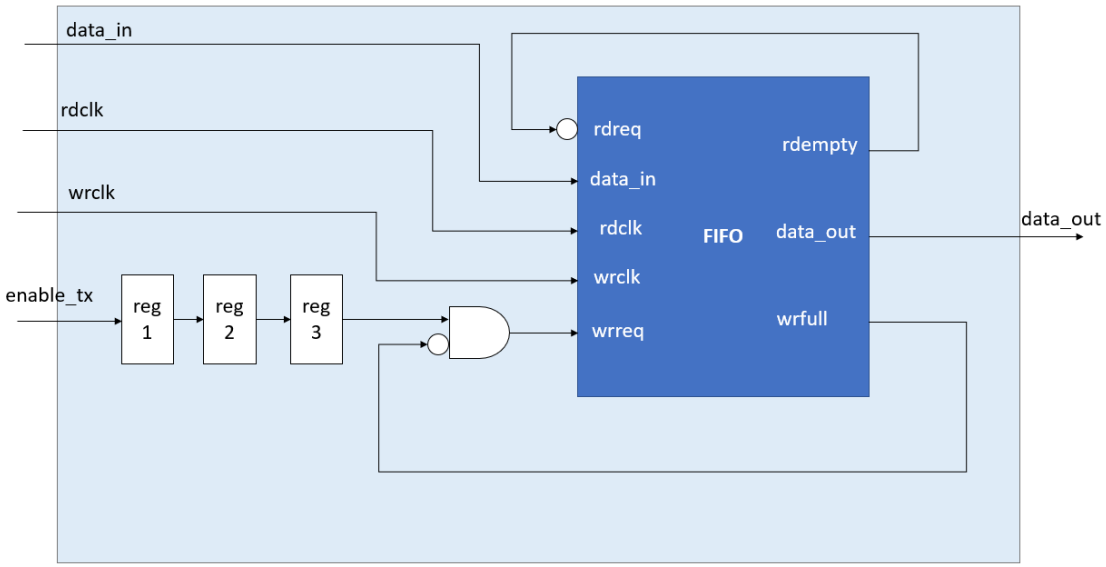


Figura 2.52 Esquema DATA_FIFO.

Para implementar la FIFO se utiliza la IP proporcionada por Altera. Se selecciona la DCFIFO con un ancho de 16 datos de 2 bits.

La escritura de datos en la FIFO comienza cuando se activa la señal de petición de escritura wrreq. Esta señal se asigna a la salida de una puerta and de enable_tx retrasada 3 ciclos y la inversa de wrfull. Para evitar corrupción de datos, según las notas de aplicación de Altera, no debe activarse

wrreq mientras wrfull esté activa. En cuanto a enable_tx, es la señal que activa el transmisor, esta cruza ambos dominios de reloj, por eso se registra múltiples veces como sincronizador.

Rdreq es la señal de petición de lectura y está activa siempre que rdempty sea cero. Por lo que estará activa mientras allá datos en la FIFO. De esta forma se asegura que se leen todos los datos de la cola, aunque enable_tx haya sido desactivada un par de ciclos antes de terminar la lectura de datos.

2.3.6 Control del sistema

El control se implementa como una máquina de estados de Moore en la que cada etapa corresponde con uno de los estados del sistema.

La etapa inicial Init_State, pone a 0 todas las señales excepto Init_state_indicator, que se activa para indicar al usuario que el sistema está listo para comenzar.

Cuando se pulsa el botón de inicio, se activa la señal start_calibration_button y la máquina de estados pasa a la etapa Start_Calibration. Aquí se activa la señal start_calibration que se envía a la máquina de estados de CALIBRATION_CONTROL para que inicie la fase de calibración.

Un ciclo de reloj más tarde se pasa a la etapa Calibration, donde se activa la señal calibration_state_indicator para comunicar al usuario que se está trabajando en la calibración. Se permanece en esta etapa hasta que se activa desde el control de calibración la señal de calibration_end.

Finalizada la calibración, la siguiente etapa es Wait_Set_PH, donde se activa la señal set_ph que comunica a la máquina de estados de incremento de fase que se desfase el reloj hasta la fase obtenida en la etapa de calibración.

Una vez se recibe la señal set_ph_done que indica que se ha realizado el desfase, se pasa a la etapa Wait_Input_BER. Se activa la señal wait_input_ber_indicator que avisa que se está esperando un input para pasar a la siguiente fase. Esta puede activarse mediante un botón o un swich que activan las señales start_ber_swich o start_ber_button que permiten pasar a la siguiente etapa.

En Start_BER, se activa la señal start_ber y tx_mode, la primera indica al control de la fase de cálculo del BER que comience a operar y la segunda cambia el transmisor al modo BER. Tras un ciclo de reloj se cambia a la siguiente etapa.

En la etapa BER, se activa la señal ber_state_indicator, que comunica que se está trabajando en la fase del cálculo del BER.

Si ocurre un timeout en la fase del BER, se indica con la señal timeout_indicator y se reinicia la fase del BER. En caso contrario, si el cálculo del BER se realiza con éxito, se recibe la señal del BER_end desdel BER_CONTROL y se pasa la última etapa.

En Finish se indica que el proceso ha finalizado y todos los registros están actualizados con la señal finish_indicator, si se activa finish_button se vuelve a la etapa inicial.

En la Figura 2.53 se muestran las etapas del sistema. Las señales mantienen los valores anteriores si no se indica lo contrario.

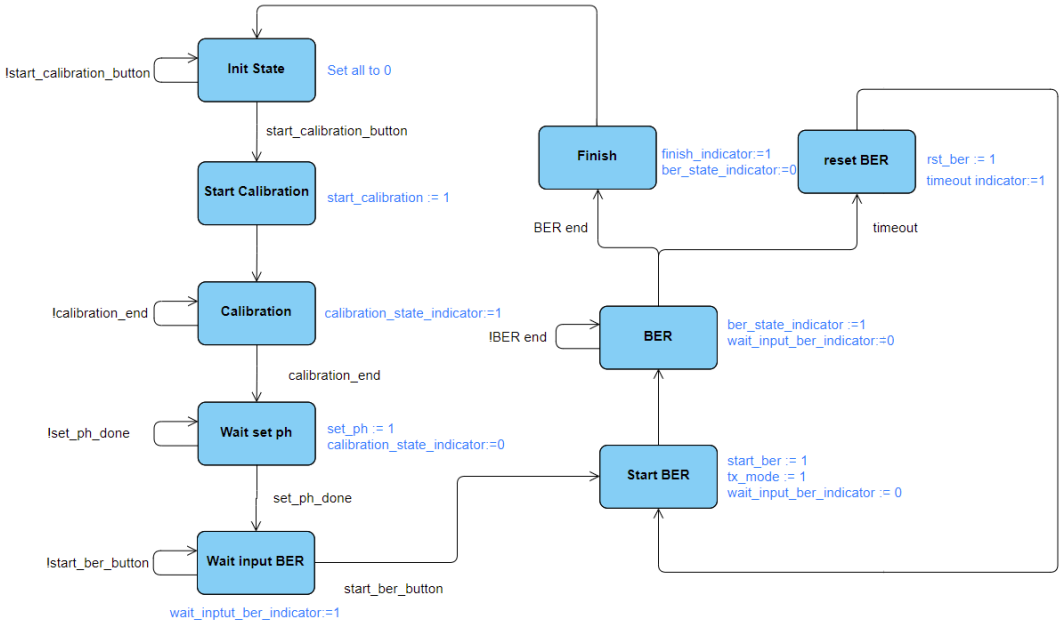


Figura 2.53 Etapas de MAIN_CONTROL.

Se utilizan las siguientes entradas y salidas:

- clk: Entrada de reloj.
- rst: Entrada de reset.
- start_calibration_button: Entrada de interfaz para iniciar el proceso de calibración.
- start_ber_swich: Entrada de interfaz para iniciar el proceso del cálculo del BER.
- start_ber_button: Entrada de interfaz para iniciar el proceso del cálculo del BER.
- finish_button: Entrada de interfaz para finalizar el proceso.
- set_ph_done: Entrada que indica que se ha realizado el cambio de fase.
- timeout: Entrada que indica que ha habido un timeout.
- ber_end: Entrada que indica que se ha finalizado el cálculo del BER.
- calibration_end: Entrada que indica que se ha finalizado el proceso de calibración.
- init_state_indicator: Salida de interfaz para indicar que se está en la etapa inicial del sistema.
- calibration_state_indicator: Salida de interfaz que indica que el sistema está realizando la etapa de calibración.
- wait_input_ber_indicator: Salida de interfaz que indica que se está esperando a que el usuario realice la acción para pasar a la fase del BER.
- ber_state_indicator: Salida de interfaz para indicar que el sistema está realizando el cálculo del BER.
- finish_indicator: Salida de interfaz que indica que los procesos han finalizado.
- timeout_indicator: Salida de interfaz que indica que ha habido un timeout.
- set_ph: Salida que indica que se desfase el reloj hasta la fase óptima.
- start_ber: Salida para comenzar el cálculo del BER en su máquina de estados correspondiente.
- start_calibration: Salida para comenzar la fase de calibración en su máquina de estados correspondiente.
- tx_mode: Salida para indicar el modo de transmisión de datos.

En la Tabla 2.15 se muestran las señales que se activan en cada etapa. Si no se especifica, la señal mantiene el valor de la etapa anterior.

Init_State		Todo a 0 excepto: Init_state_indicator = 1	
Start_Calibration		Init_state_indicator = 0 Start_calibration = 1	
Calibration		Start_calibration = 0 Calibration_state_indicator = 1	
Wait_Input_BER		Calibration_state_indicator = 0 Wait_input_ber_indicator = 1	
Start_BER		Wait_input_ber_indicator = 0 Start_ber = 1 Tx_mode = 1	
BER		Start_ber = 0 Ber_state_indicator = 1	
Reset_BER	Ber_state_indicator = 0 Timeout_indicator = 1	**Finish**	Ber_state_indicator = 0 Finish_indicator = 1

Tabla 2.15 Activación de señales en cada etapa MAIN_CONTROL.

2.3.7 Test RTL sistema completo

Se realiza un banco de pruebas con el sistema completo. Tanto para las señales de entrada del test, como para contrastar los resultados obtenidos, se utilizan los ficheros de texto generados para el test de los bloques de calibración y cálculo del BER.

En el test se generan las señales de necesarias para avanzar la etapa del control del sistema cuando se activa previamente la señal que indica que se puede continuar con el proceso. Por ejemplo, cuando se activa la señal que indica que se ha finalizado con la etapa de calibración, el test espera unos ciclos y activa la señal que permite continuar a la etapa del cálculo del BER.

Se realizan pruebas modificando distintos parámetros como en los test de los bloques de calibración y cálculo del BER vistos anteriormente. Se observa cómo los resultados obtenidos en el hardware coinciden con los de Matlab.

Finalmente, se modifica el test para comprobar que la transmisión funciona correctamente. Se conecta la salida del transmisor directamente a la entrada del sistema, que corresponde al puerto de escritura de la FIFO. Se comprueba como se realiza la fase de calibración correctamente. En la etapa del BER, en el visor de ondas se puede ver que se detecta el preámbulo y se queda en esta etapa contando muestras constantemente sin detectar errores. Esto ocurre porque al estar la salida y la entrada directamente conectadas, tanto la señal de entrada como la generada internamente para calcular el BER coinciden.

Capítulo 3. Interfaz con Matlab

3.1 Creación del componente avalon_main_block

Para realizar la escritura y lectura de registros del sistema desde Matlab, se utiliza el puerto Ethernet al que se accede desde el hard processor system de la tarjeta DE10-Standard. Por lo tanto, es necesario establecer la comunicación entre el HPS de la FPGA y el sistema diseñado. Para ello, se crea un periférico en el Platform Designer que contenga el hardware siguiendo la metodología descrita en el documento de diseño y verificación de periféricos [12]. Este periférico se compone de tres módulos:

- Avalon_main_block: Top de la jerarquía en el que se instancian el interfaz del bus Avalon y el sistema diseñado. En sus puertos están las líneas del interfaz Avalon-MM slave y las salidas que se conectan a los LEDs para indicar el estado del sistema. Además, el sistema funciona con una señal entre 1 y -1 pero los datos transmitidos y recibidos se codifican con 1 y 0. En este bloque se adapta la señal a la entrada y salida.
- Avalon_slave_MM_interface: Conecta el sistema con el bus Avalon-MM. Esto permite que el hardware se comporte como un periférico Avalon-MM slave, permitiendo la comunicación con el microprocesador realizando escrituras y lecturas de memoria.
- Main_block: Hardware diseñado.

Uno de los objetivos al crear el componente, era poder trabajar con la tarjeta desde un escritorio remoto. Por ello, se configura el periférico de forma que las señales de entrada del hardware se controlen desde registros escritos desde el microprocesador. A continuación, se describen los registros.

Signal_reg: Registro de escritura en la dirección 0, contiene las señales que permiten que el sistema avance de una etapa a otra. Cada bit corresponde con un swich o botón de la interfaz.

Señal	**registro**
Rst_pll	000001
Rst_ber	000010
Start_calibration	000100
Start_ber	001000
Finish	010000
Rst	100000

Tabla 3.1 Configuración registro signal_reg.

Input_config_reg: Registro de escritura en la dirección 1, contiene la configuración con el puerto GPIO que se selecciona como entrada del sistema. Para realizar las pruebas se utilizan 3 cables de longitud distinta para simular distintos medios de transmisión. Con este registro se selecciona que cable se utiliza para no tener que recompilar.

Cable	**registro**
Corto	00
Medio	01
Largo	10

Tabla 3.2 Configuración registro input_config_reg.

Pll_select_reg: Registro de escritura en la dirección 2, contiene la información de que par de relojes del pll se va a utilizar.

Par de relojes	**registro**

outclk_0f y outclk_0d	00
outclk_1f y outclk_1d	01
outclk_2f y outclk_2d	10
outclk_3f y outclk_3d	11

Tabla 3.3 Configuración registro pll_select_reg.

Indicator_reg: Registro de lectura en la dirección 0, contiene la información del estado del sistema. Coincide con los LEDs del interfaz.

Estado	**registro**
Estado inicial	000001
Calibración	000010
Espera al input para pasar al BER	000100
Cálculo del BER	001000
Finalizado	010000
Timeout	100000

Tabla 3.4 Configuración registro indicator_reg.

Tx_bits_cnt_LSB: Registro de lectura en la dirección 1. Como el contador es mayor de 32 bits, se divide en dos registros. Este contiene la información del contador de muestras en el intervalo de 0 a 31 bits.

Tx_bits_cnt_MSB: Registro de lectura en la dirección 2, contiene la información del contador de muestras en el intervalo de 32 hasta el máximo tamaño del contador.

Error_n: Registro de lectura en la dirección 3, almacena el número de errores contabilizados en la fase del cálculo del BER.

Selected_phase: Registro de lectura en la dirección 4, informa de la fase seleccionada en la etapa de calibración.

En la Figura 3.1 se muestra la estructura del periférico.

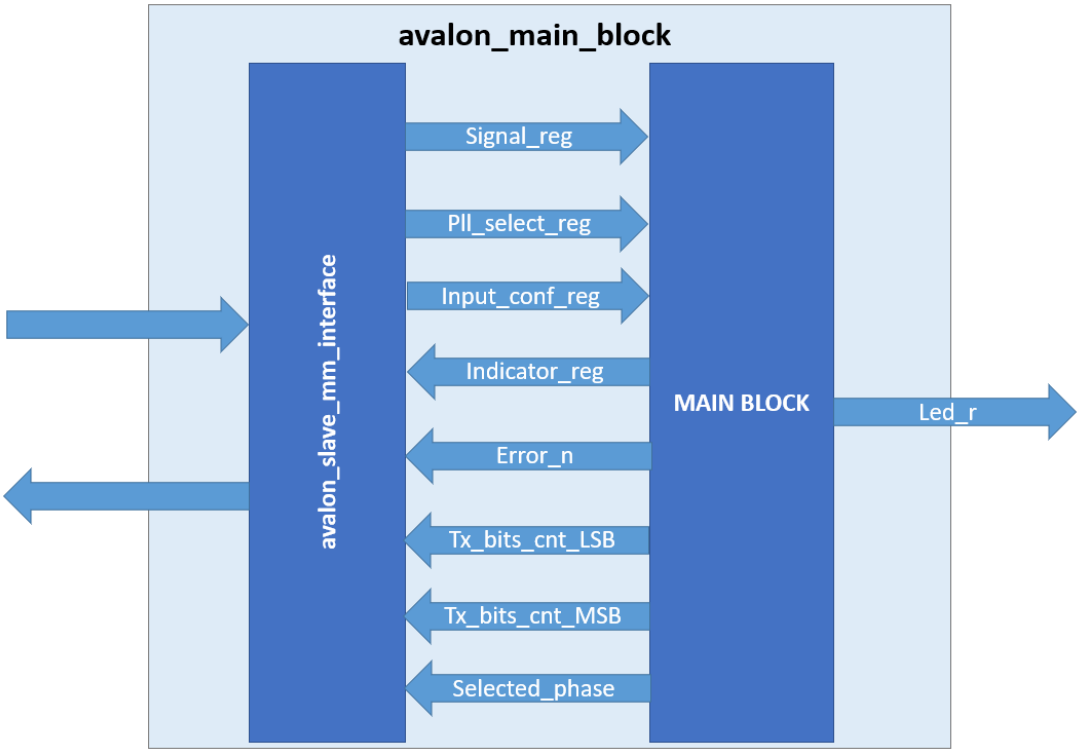


Figura 3.1 Estructura avalon_main_block.

3.2 Sistema en Platform Designer

Para la integración en Platform Designer, el sistema cuenta con las siguientes interfaces:

- Interfaz Clock Input para el reloj de referencia del PLL.
- Interfaz Reset input para el reset del sistema.
- Interfaz Avalon Memory Maped Slave para la comunicación a través del bus Avalon-MM.
- Intefaz Conduit: Para las líneas de salida conectadas a los LEDs.

Se parte del proyecto DE10_Standard_GHRD que se explica en el manual de usuario DE10-Standard [13]. Este proyecto contiene los siguientes componentes:

- ARM Cortex-A9 MPCore HPS.
- 4 botones de entrada.
- 10 switches de entrada.
- 10 LEDs.
- 64Kb de memoria on-chip.
- JTAG a los puentes del Avalon máster.
- Captura de interrupciones.
- System ID.

El mapa de memoria de los periféricos del proyecto se muestra en la Figura 3.2.

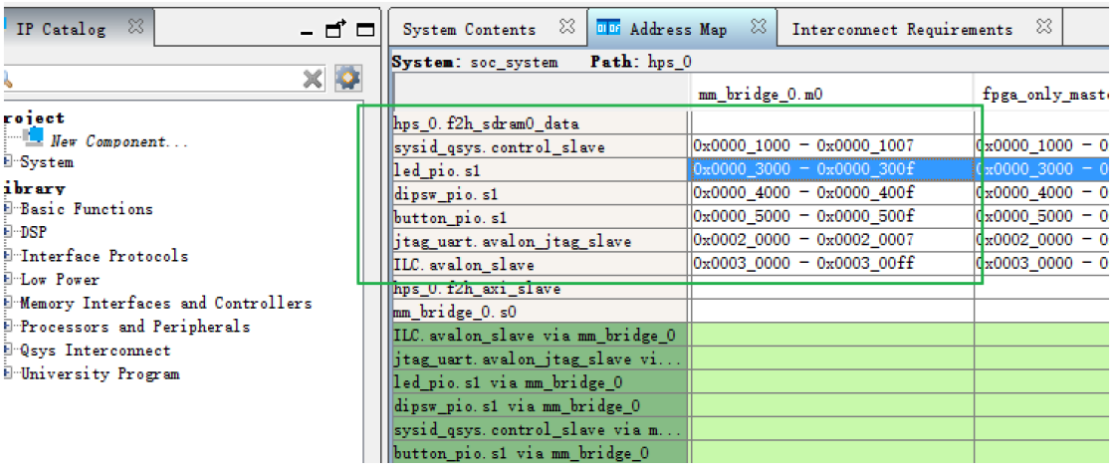


Figura 3.2 Mapa de memoria de los periféricos del proyecto DE10_Standard_GHRD. Fuente [13].

En la dirección 0, se introduce el componente Avalon_main_block diseñado. Se conecta la entrada de reloj a la fuente clk_0, el reset al reset de clk_0y el Avalon mapped memory slave al puente Avalon-MM 0 tal y como se muestra en la Figura 3.3.

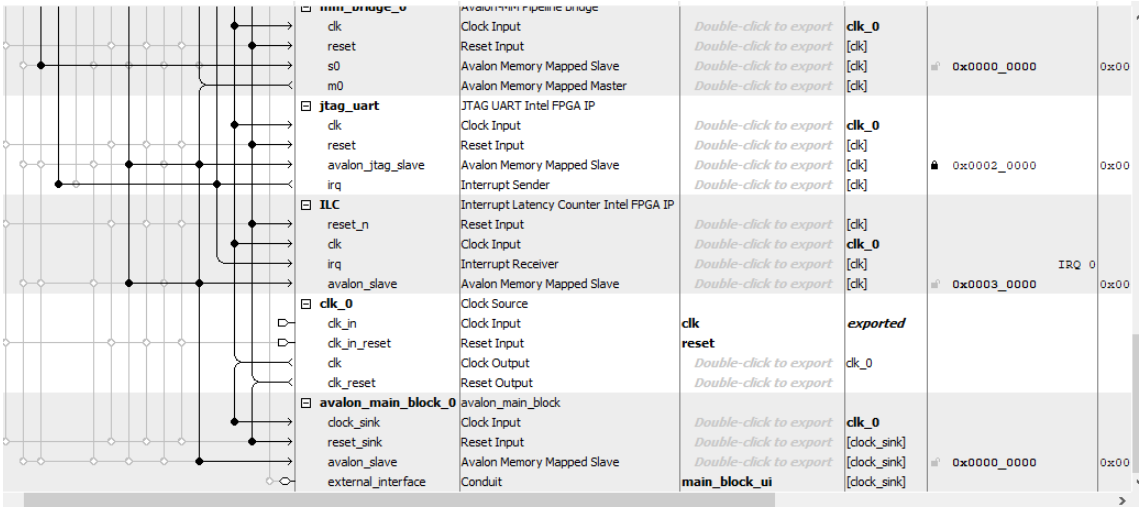


Figura 3.3 Integración de avalon_main_block en el platform designer.

3.3 Comunicación entre Matlab y el Microprocesador

Para implementar el software que se encarga de la comunicación de Matlab con el microprocesador, se utiliza como referencia el proyecto HPS_FPGA_LED que se explica en el manual DE10-Standard [13].

El software se implementa en Visual Studio. En este programa, se realiza el mapeado de la dirección del periférico de los LEDs en memoria virtual, a la que se puede acceder desde el software. Se mapea la dirección base de la región de periféricos en una dirección virtual virtual_base. En el manual se explica como la dirección virtual del periférico al que se quiere acceder, puede calcularse sumando dos offsets a virtual_address. Por una parte el offset de la dirección del puente HPS-to-FPGA AXI bus relativo a la dirección del HPS. Por otra, el ofsset de la dirección del periférico relativo al puente HPS-to-FPGA AXI bus. Como el periférico Avalon_main_block se encuentra en la dirección cero del puente, solo es necesario añadir el primer offset.

Con el periférico mapeado, el siguiente paso es implementar la comunicación entre el ordenador y el microprocesador de la FPGA. Para ello se implementa en el software un server TCP a partir de un proyecto encontrado en la web [14].

Una vez realizada la conexión, el programa espera constantemente la recepción de mensajes desde el cliente y los almacena en un buffer. Se codifica el mensaje de forma que, la primera posición del buffer corresponde con el tipo de acceso a la memoria del periférico. Si es un 0 se realiza una operación de escritura, y si es un 1 de lectura. La segunda posición del buffer corresponde a la dirección del periférico. En el caso de que sea una operación de escritura, en la tercera posición se envía le dato a escribir en el registro direccionado en la segunda posición del buffer. En la Tabla 3.5 se muestra el formato del mensaje.

Buffer[0]		Buffer[1]	Buffer[2]
Tipo de operación	0: Lectura 1: Escritura	Dirección del interfaz Avalon slave del periférico	Dato a escribir

Tabla 3.5 Formato del mensaje para la comunicación entre Matlab y el periférico.

En Matlab, se crea un script donde se almacenan en variables los distintos mensajes que se utilizan en la escritura o lectura del periférico. Por ejemplo, para iniciar la fase de calibración, se almacena en init_cal el vector [1 0 4]. Finalmente, se establece la conexión con el server TCP utilizando la función tcpclient descrita en el centro de ayuda de Matlab [15]. Con las funciones de escritura y lectura se realizan las operaciones que se requieran.

Para comenzar la operación del sistema, se siguen los siguientes pasos:

- Carga de mensajes y conexión con el servidor.
- Operación de escritura con el reloj que se vaya a utilizar.
- Operación de escritura con el mensaje de reset del PLL.
- Operación de escritura con el mensaje de reset del sistema.
- Operación de escritura con el mensaje de inicio de calibración.
- Mediante los LEDs o realizando una operación de lectura del registro indicator_reg, comprobar que el sistema está esperando input para pasar a la fase del cálculo del BER.
- Operación de escritura con el mensaje de inicio de la etapa del BER.

Capítulo 4. Testeo del prototipo

4.1 Pruebas con SignalTap

Se carga el prototipo en la tarjeta y se realizan pruebas con distintas configuraciones para comprobar el funcionamiento del sistema. Se utiliza el SignalTap para poder observar la evolución de las señales y se confirma que el sistema se comporta como se espera.

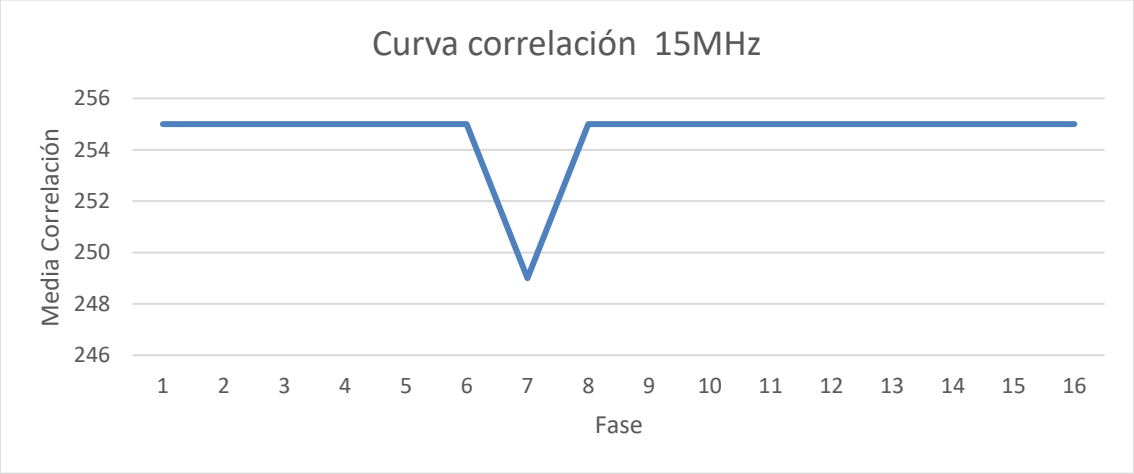
Se comprueba cómo evoluciona la curva de correlación media por fase en función de la frecuencia. Como base de referencia se utiliza la siguiente configuración:

- Secuencia de calibración de 255 bits.
- 16 fases.
- 4 contadores en cascada de 13 bits.
- 10 ceros tras la secuencia de calibración.
- 1024 envíos de la secuencia de calibración por fase.

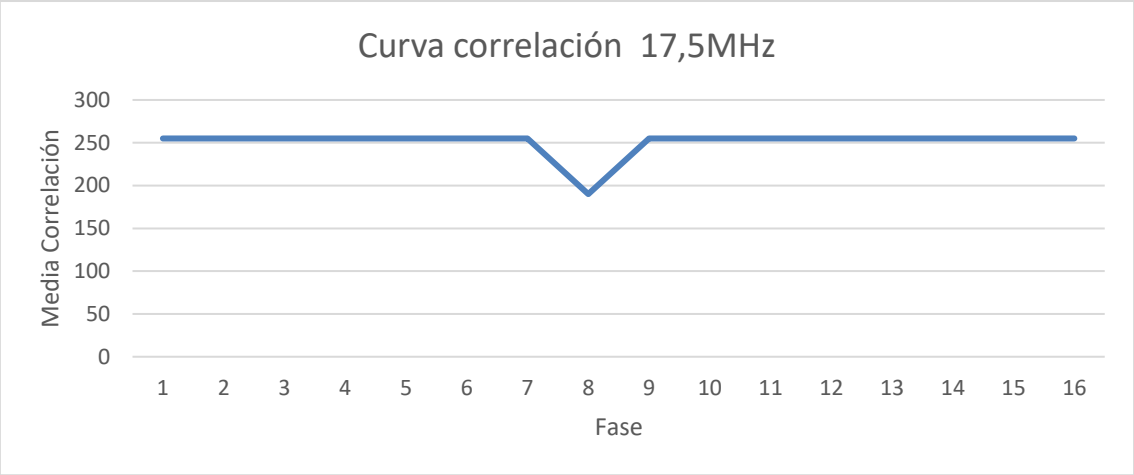
Para comprobar que el PLL genera la frecuencia correctamente, se cuenta el número de muestras en 10 segundos y se divide por la frecuencia del reloj seleccionado. Si el resultado es aproximado a 10, se está generando la frecuencia correctamente.

Se comprueba que la selección de fase se está realizando correctamente introduciendo la curva obtenida en la parte de Matlab que se encarga de la selección de fase y viendo que coincide con la seleccionada por el hardware.

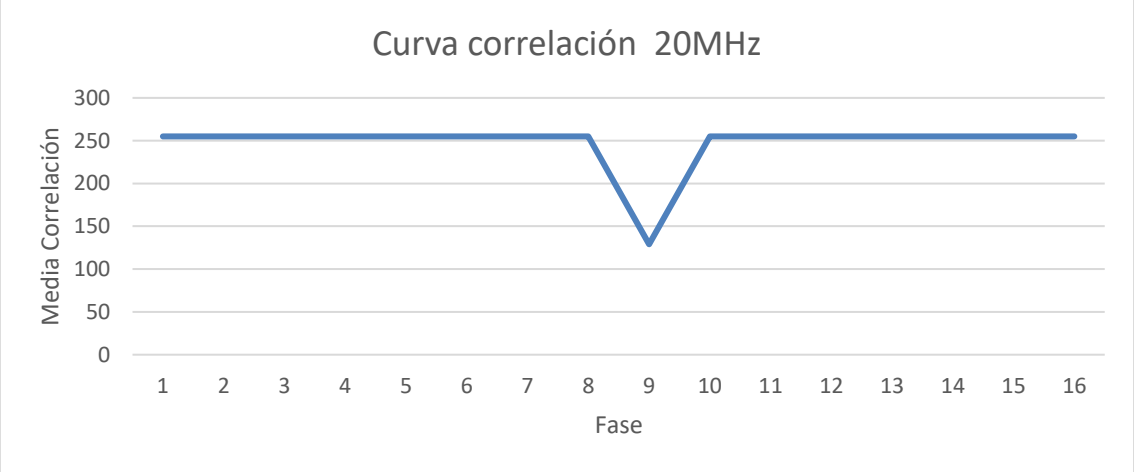
Para frecuencias inferiores a 15 MB/s todas las fases tienen una correlación máxima, a partir de 15 MB/s comienzan a aparecer fases con pérdidas como se observa en Figura 4.1.



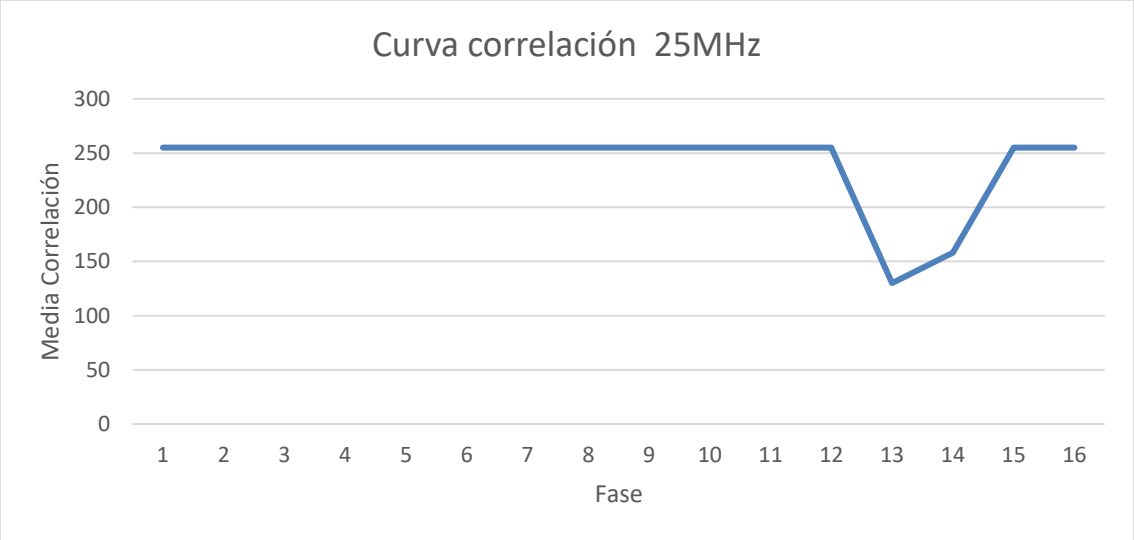
(a)


Curva correlación 17,5MHz
Media Correlación
300
250
200
150
100
50
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Fase

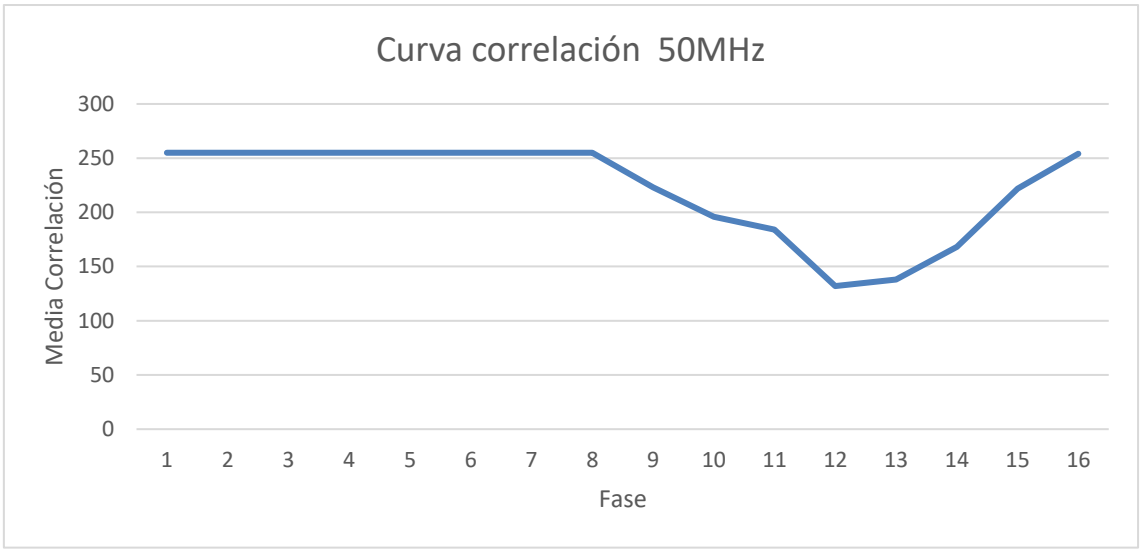
(b)


Curva correlación 20MHz
Media Correlación
300
250
200
150
100
50
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Fase

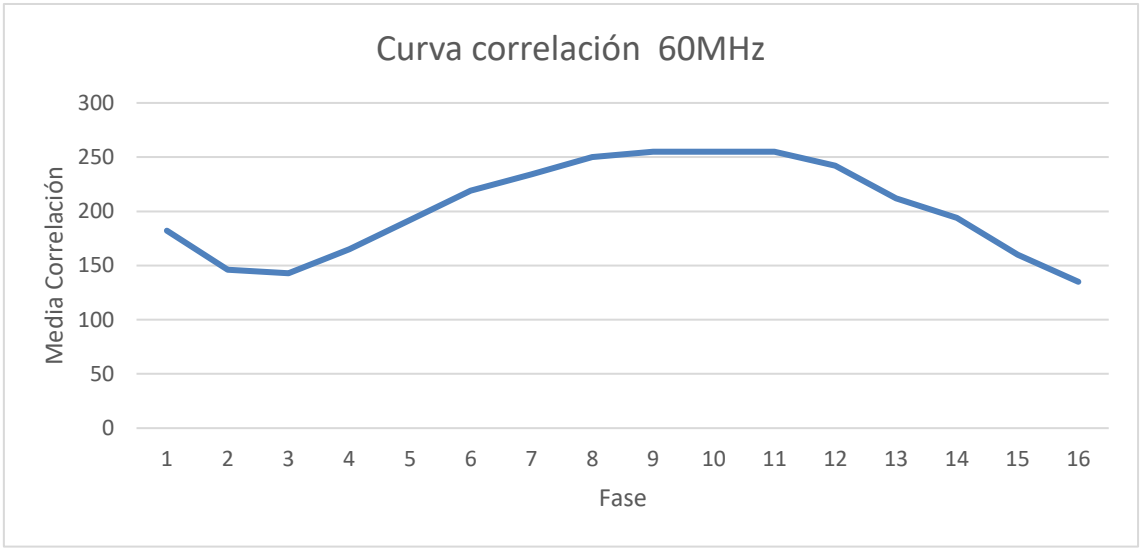
(c)


Curva correlación 25MHz
Media Correlación
300
250
200
150
100
50
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Fase

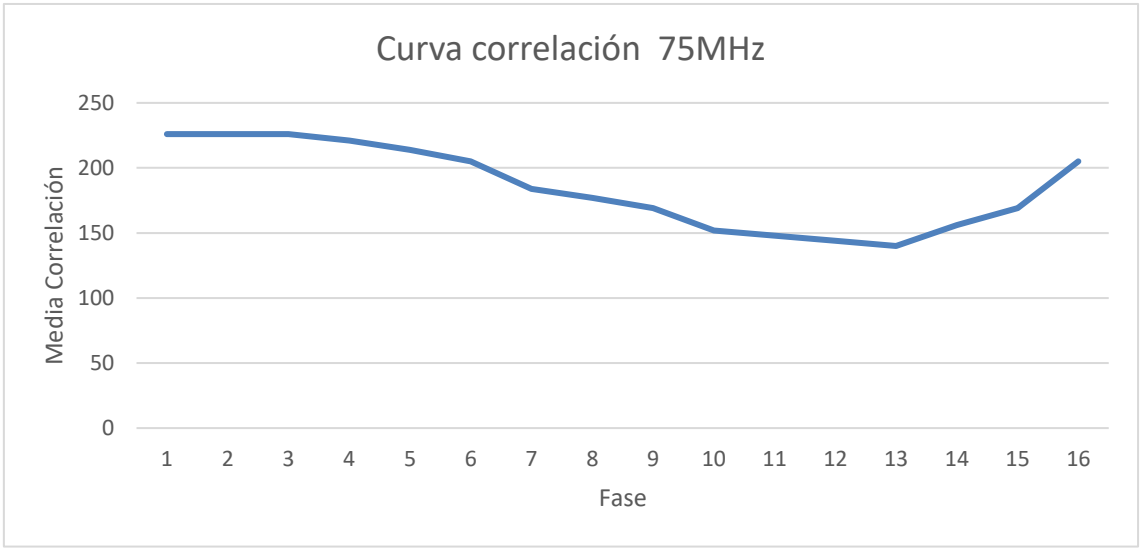
(d)


Curva correlación 50MHz
Media Correlación
300
250
200
150
100
50
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Fase

(e)


Curva correlación 60MHz
Media Correlación
300
250
200
150
100
50
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Fase

(f)

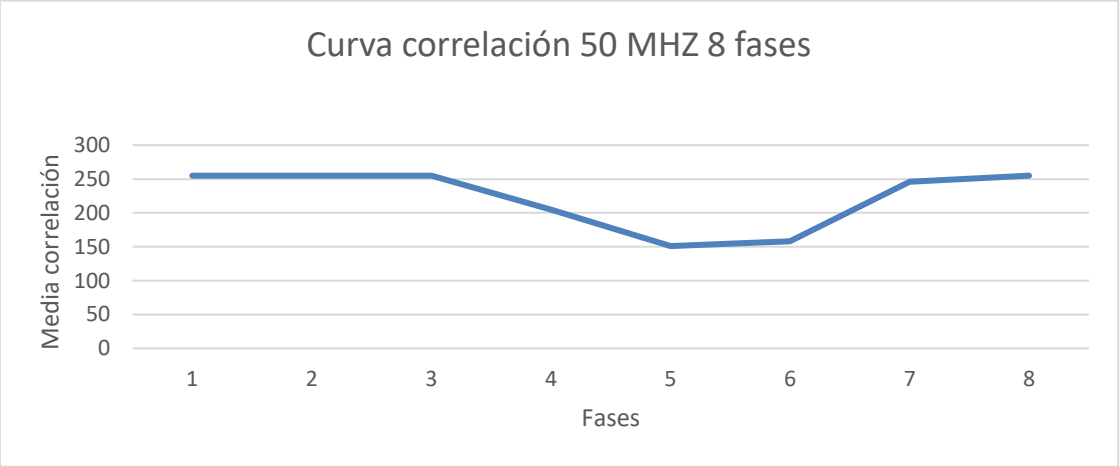

Curva correlación 75MHz
Media Correlación
250
200
150
100
50
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Fase

(g)

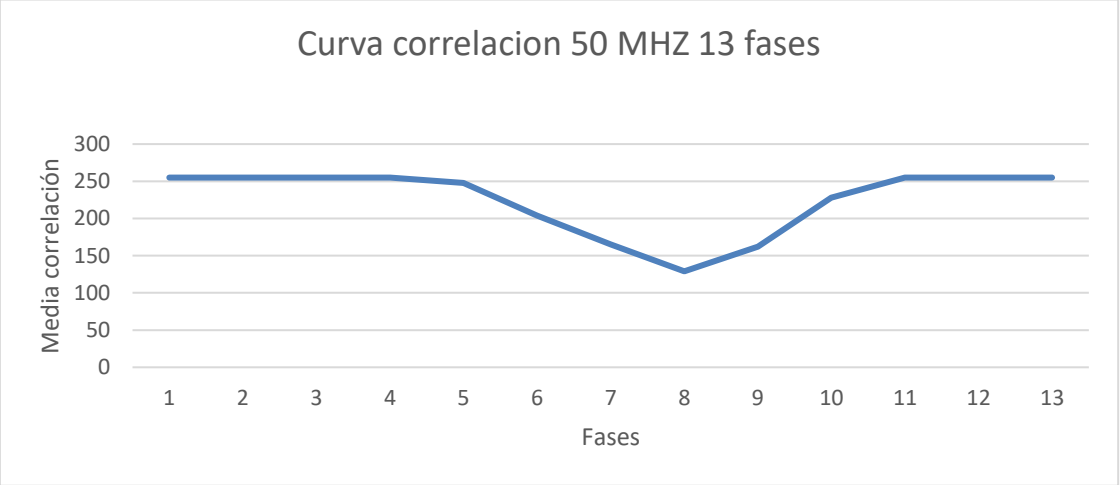
Figura 4.1 Gráficas de correlación a distintas frecuencias. a 15 MB/s. b 17,5 MB/s. c 20 MB/s. d 25 MB/s. e 50 MB/s. f 60 MB/s. g 70 Mb/s.

A medida que incrementa la frecuencia, aparecen más fases con pérdidas estrechándose el número de fases válidas. Llegados a los 75 MB/s ninguna de las fases consigue una correlación máxima.

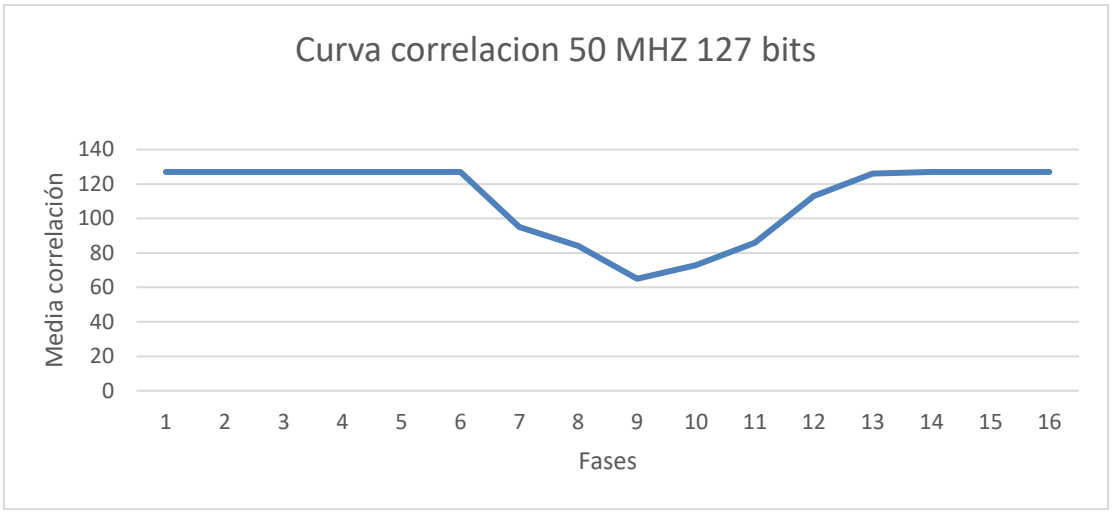
Se realizan test modificando los parámetros de configuración, los resultados se muestran en la Figura 4.2.



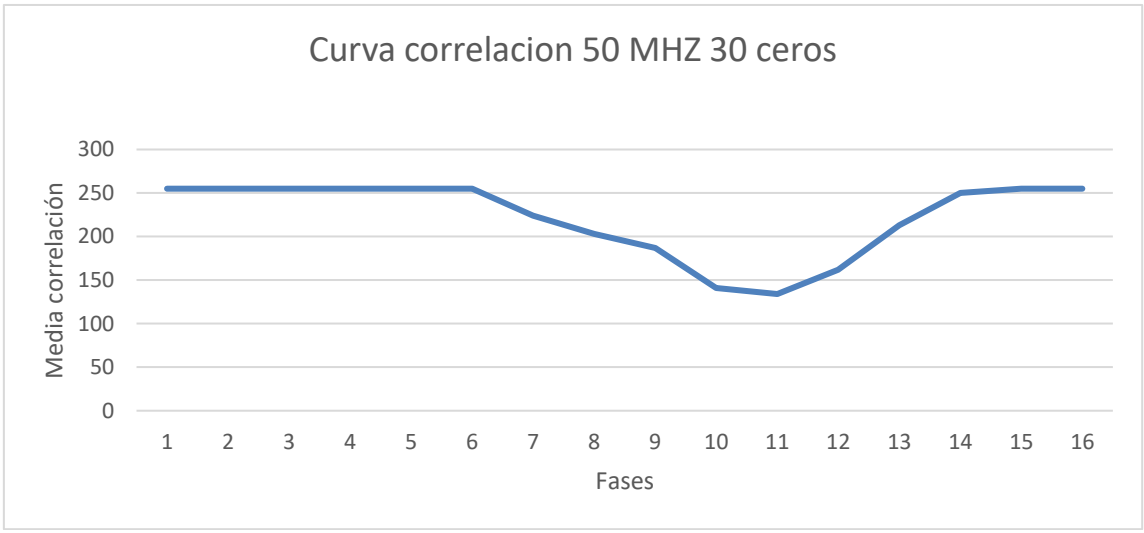
(a)



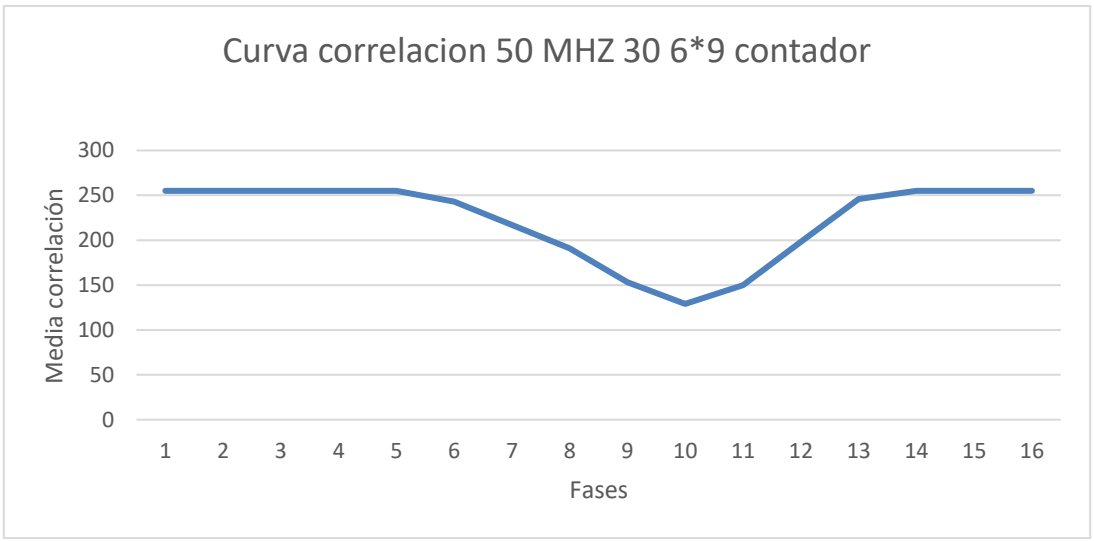
(b)



(c)



(d)



(e)

Figura 4.2 Pruebas de calibración con distintos parámetros. a 8 fases. b 13 fases. c secuencia de 127 bits. d 30 ceros tras la secuencia. e contador de muestras de 6*9.

Con el cable largo seleccionado, se realizan pruebas para la etapa del cálculo del BER. Experimentalmente se comprueba que hasta los 50 MB/s, el BER es de 0.

Se realizó una prueba dejando el sistema en la fase del BER a 50 MB/s desde las 20:34 hasta las 10:06 del día siguiente. No ocurrió ningún error de transmisión en 2435862483082 muestras, esta cantidad de muestras equivale a unas 13 horas y 32 minutos en marcha, que coincide con el tiempo que estuvo encendido el sistema.

A partir de los 60 MB/s comienzan a aparecer errores esporádicos. A 75 MB/s el BER ronda el 9% y se alcanza el número de errores necesario para finalizar el proceso del cálculo del BER instantáneamente.

4.2 Pruebas en laboratorio

El esquema del sistema que realiza la modulación del LED y la recepción de la señal en el laboratorio se muestra en la Figura 4.3.

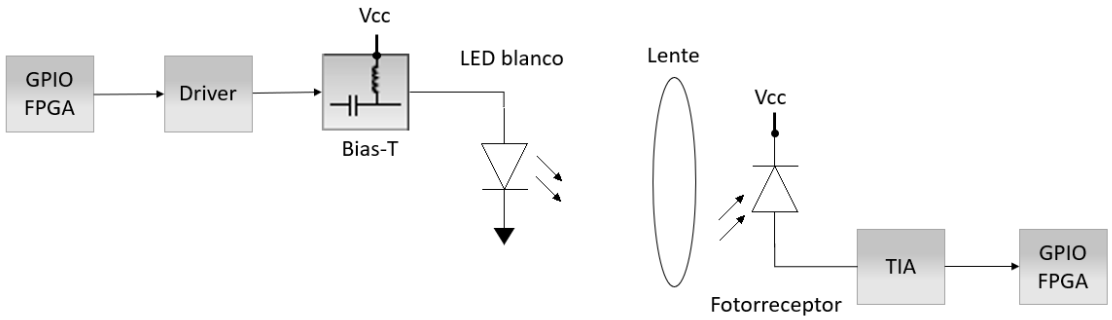
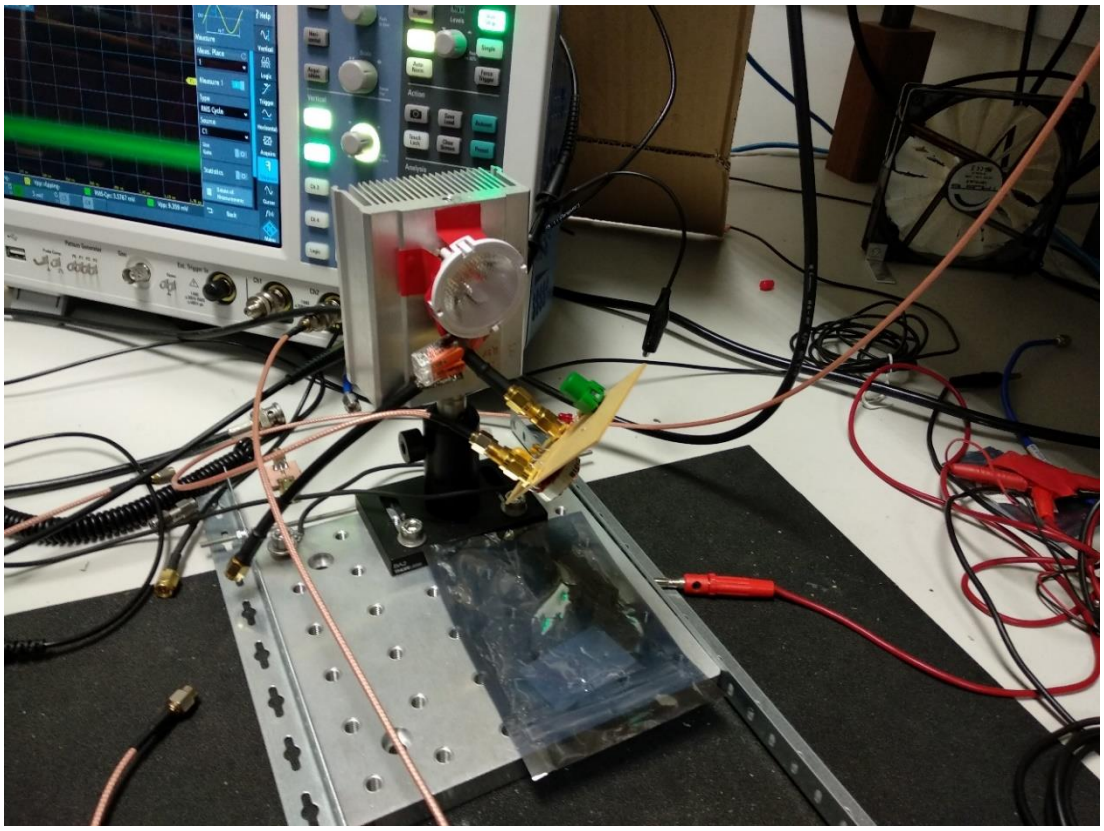


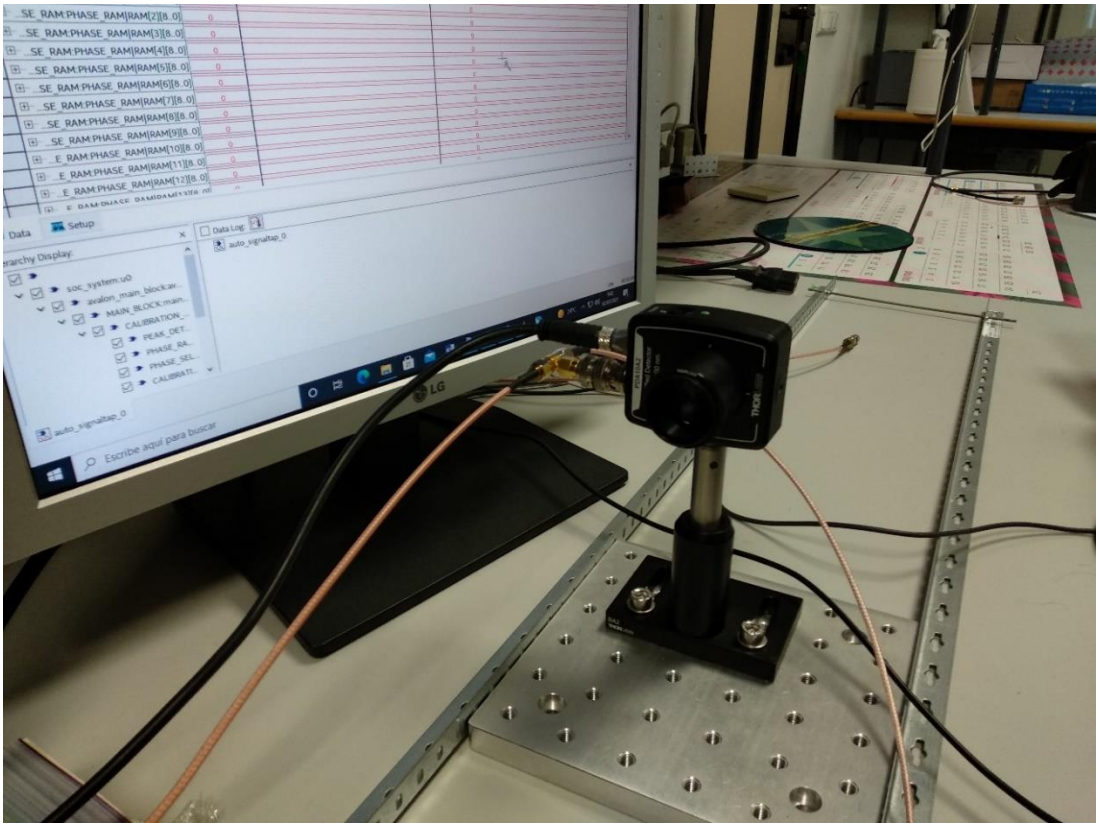
Figura 4.3 Esquema de sistema de transmisión y recepción en el laboratorio.

La señal a transmitir se obtiene de un puerto GPIO de la FPGA donde está la salida del transmisor montado en el prototipo diseñado en este proyecto. Esta pasa por un driver donde se amplifica la señal. El driver se conecta a un bias-T. El bias-T es un circuito pasivo de tres puertos en estrella. Una rama tiene un condensador, otra un inductor y la otra se conecta el LED de iluminación. A la rama del condensador se le conecta la señal a transmitir proveniente del driver y el condensador bloquea la continua. El inductor se conecta a una tensión DC para llevar al LED a su corriente de polarización, además impide el paso de señales a alta frecuencia. Con este circuito, el LED recibe un nivel de tensión continua que permite polarizar el LED, además de la señal transmitida. El LED es el modelo LZ4-40CW08-0065 de LED Engin [16]. En este modelo, hay conectados 4 LEDs en serie. Se le alimenta con una tensión de 13V y 600mA de corriente.

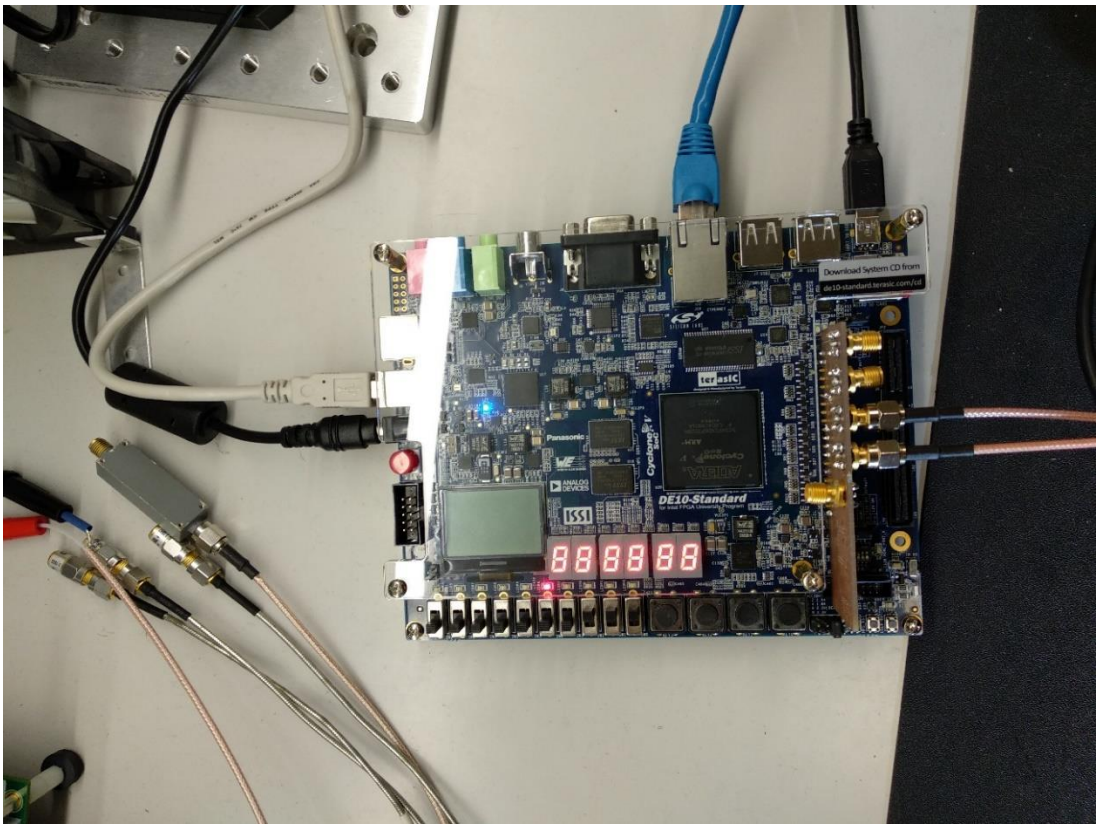
La señal transmitida por el LED se captura con un fotodiodo. La secuencia recibida se procesa en un amplificador de transimpedancia que amplifica la señal. Finalmente se conecta a un puerto GPIO de la FPGA como entrada del prototipo diseñado en este proyecto. En la Figura 4.4, se muestra el sistema montado en el laboratorio.



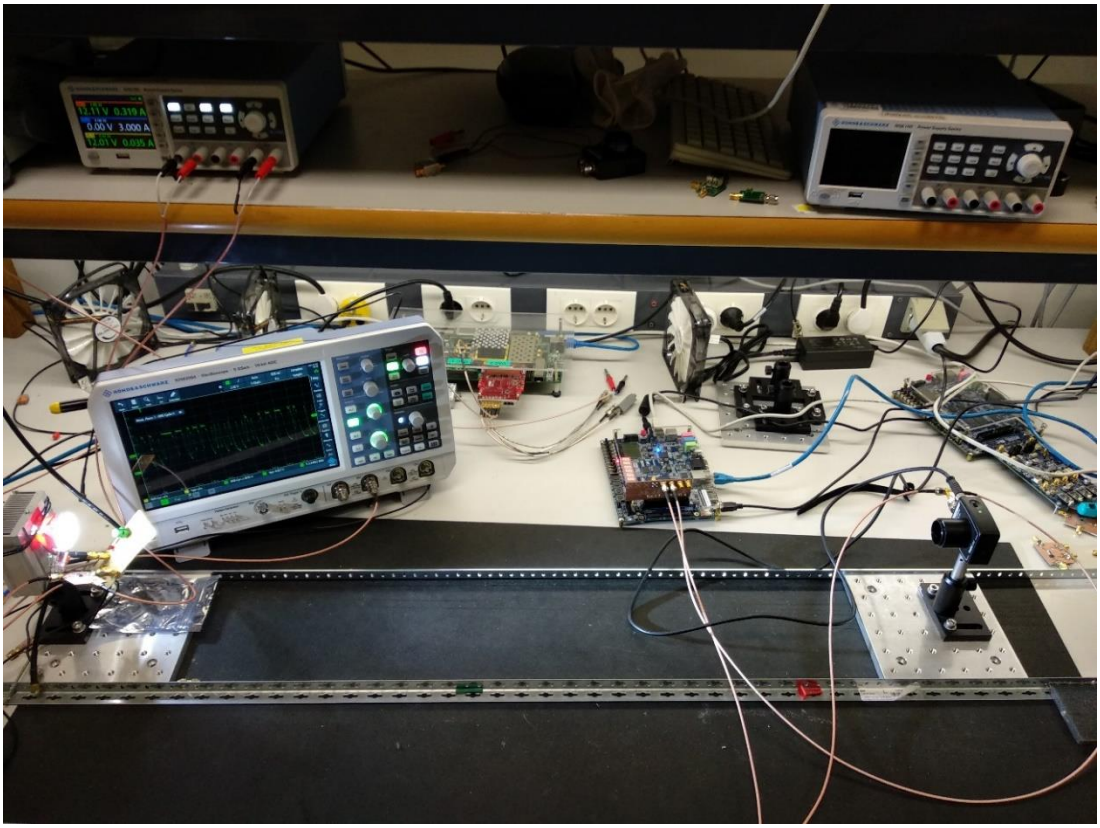
(a)



(b)



(c)



(d)

Figura 4.4 Sistema de modulación de intensidad y detección directa montado en el laboratorio. (a) LED. (b) Fotoreceptor. (c) Proptotipo montado en la placa de desarrollo DE-10 Standard. (d) Sistema completo.

El prototipo se configura con los siguientes parámetros:

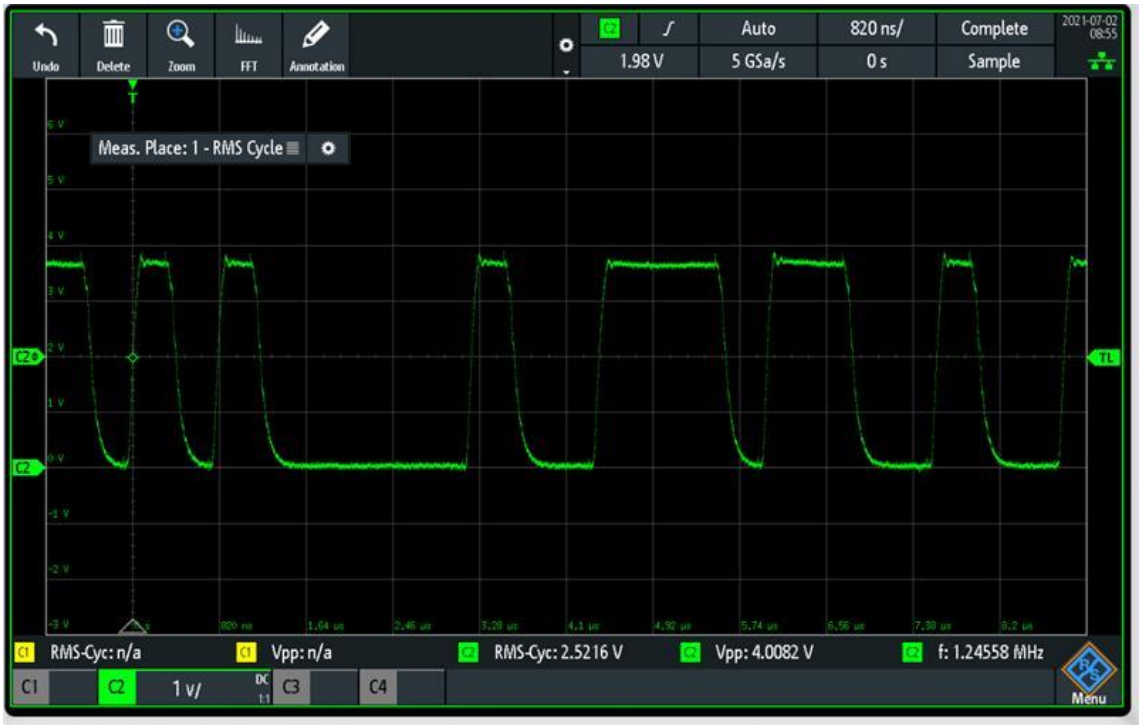
- Tamaño de la secuencia de calibración: 255 bits.
- Número de veces que se transmite la secuencia de calibración por fase para la media de correlación: 1024.
- Número de fases: 16.
- Número de errores para terminar la etapa del cálculo del BER: 100.
- Contador de muestras: 4 contadores de 13 bits = 52 bits.
- LFSR con la secuencia que se transmite en la etapa del cálculo del BER de 2^{52} bits.

El procedimiento que se sigue durante la toma de medidas es.

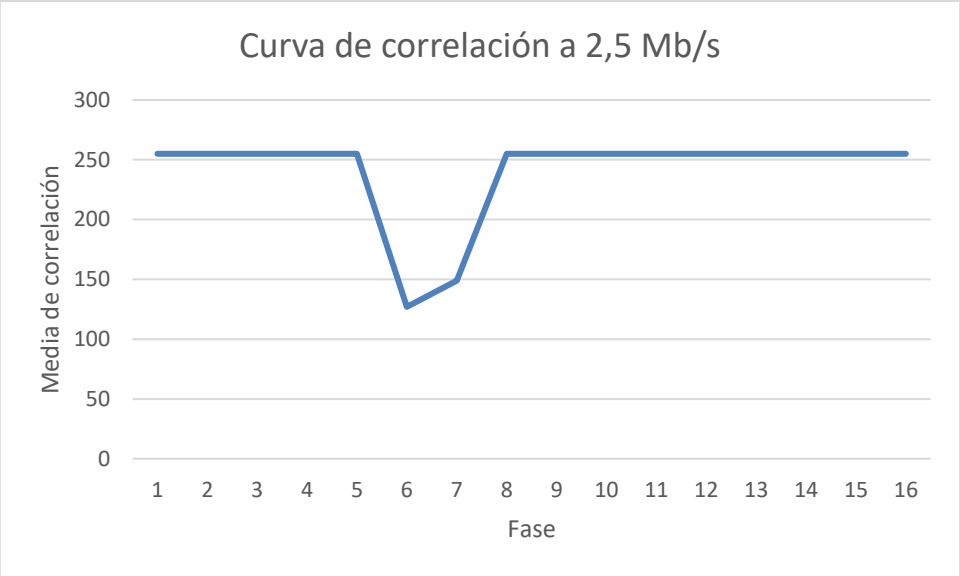
1. Configuración de frecuencia del sistema.
2. Reset del sistema.
3. Envío de las secuencias de calibración.
4. Captura de esta secuencia recibida con el osciloscopio y del valor medio de correlación en cada fase.
5. Envío de la secuencia del BER y comprobación del BER.

Todas las pruebas se realizan con una distancia entre el LED y el fotorreceptor de 175 cm. Esta distancia puede darse en una aplicación fuera de laboratorio del sistema, si por ejemplo se realiza la transmisión de señal entre una lámpara LED en el techo y un receptor sobre un escritorio.

Se realizan medidas con frecuencias entre 2.5MHz y 10 MHz. Para 2.5 Mb/s los resultados se muestran en Figura 4.5. La señal recibida es bastante limpia y muchas de las fases tienen máxima de correlación, El valor medio de correlación de cada fase se extrae con el SignalTap. La fase óptima es la 5, que coincide con la fase central de entre todas las que superan el umbral, que es 191.



(a)

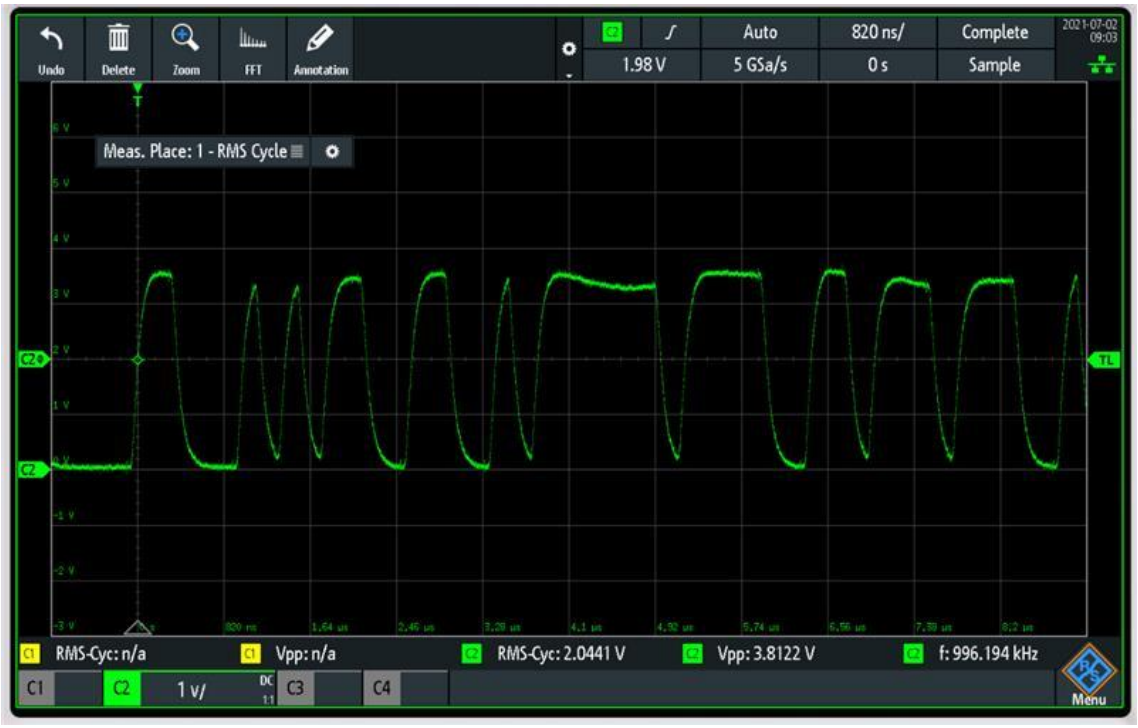


(b)

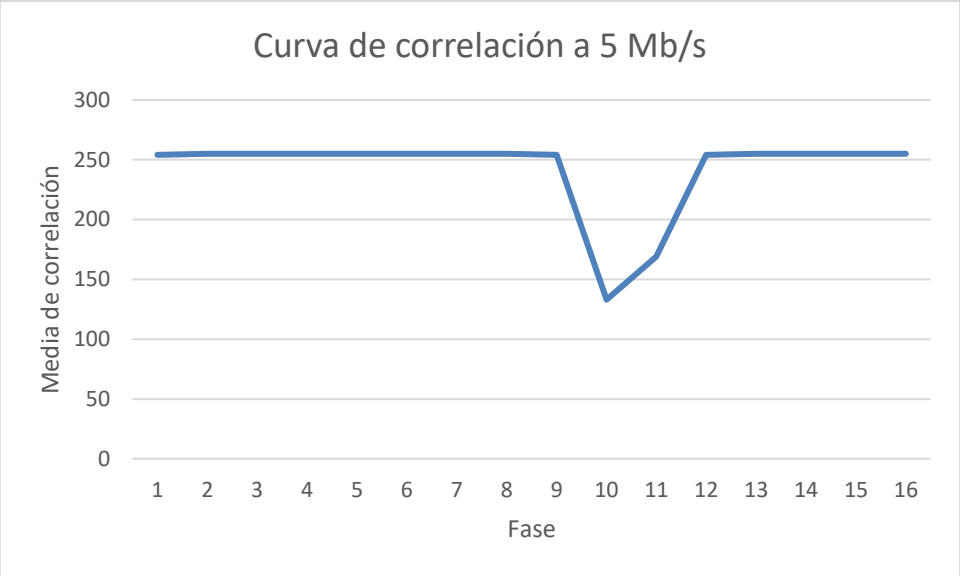
Figura 4.5 Calibración a 2.5 Mb/s. (a) secuencia de calibración capturada en el osciloscopio. (b) curva de correlación.

El cálculo del BER se detiene manualmente cuando el orden de los bits recibidos es de 10^9 sin detectar ningún error. Se divide 1 por el número de bits recibidos cuando se fuerza la parada del cálculo del BER y se puede decir que el BER es inferior a 1.7921×10^{-9}.

Para 5 Mb/s los resultados se muestran en la Figura 4.6. Se aprecia cómo la señal recibida se deforma mínimamente. Muchas de las fases tienen correlación máxima y la curva de correlación es similar a la obtenida en la transmisión a 2.5 Mb/s. La fase óptima es la 4, que coincide con la fase central de entre todas las que superan el umbral, que es 194.



(a)



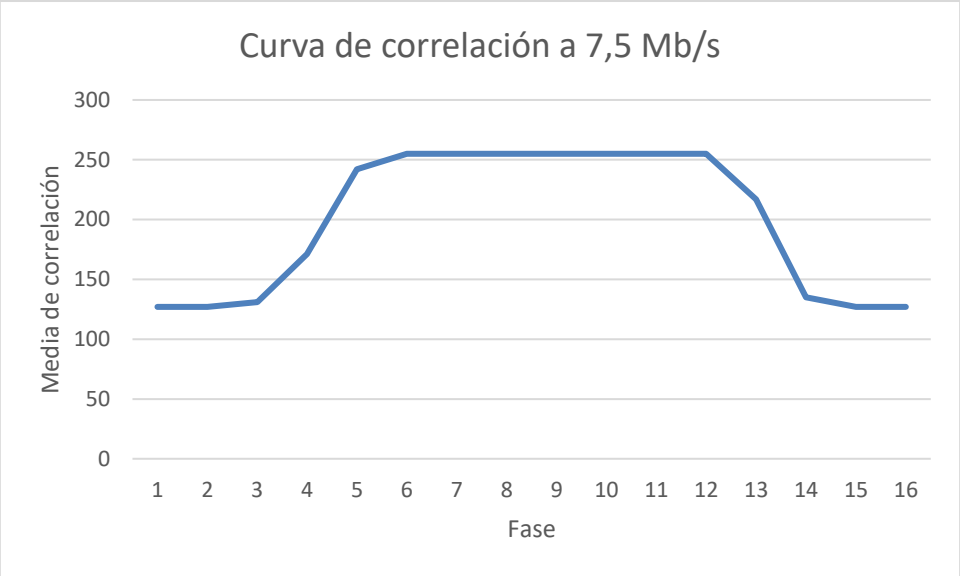
(b)

Figura 4.6 Calibración a 5 Mb/s. (a) secuencia de calibración capturada en el osciloscopio. (b) curva de correlación.

El cálculo del BER se detiene manualmente cuando el orden de los bits recibidos es de 10^9 sin detectar ningún error. Se divide 1 por el número de bits recibidos cuando se fuerza la parada del cálculo del BER y se puede decir que el BER es inferior a 1.5676×10^{-9}.

Para 7.5 Mb/s los resultados se muestran en Figura 4.7.Se aprecia como la señal comienza a perder integridad, y disminuye el número de las fases que tienen máxima correlación. La fase óptima es la 9, que coincide con la fase central de entre todas las que superan el umbral, que es 191.

(a)

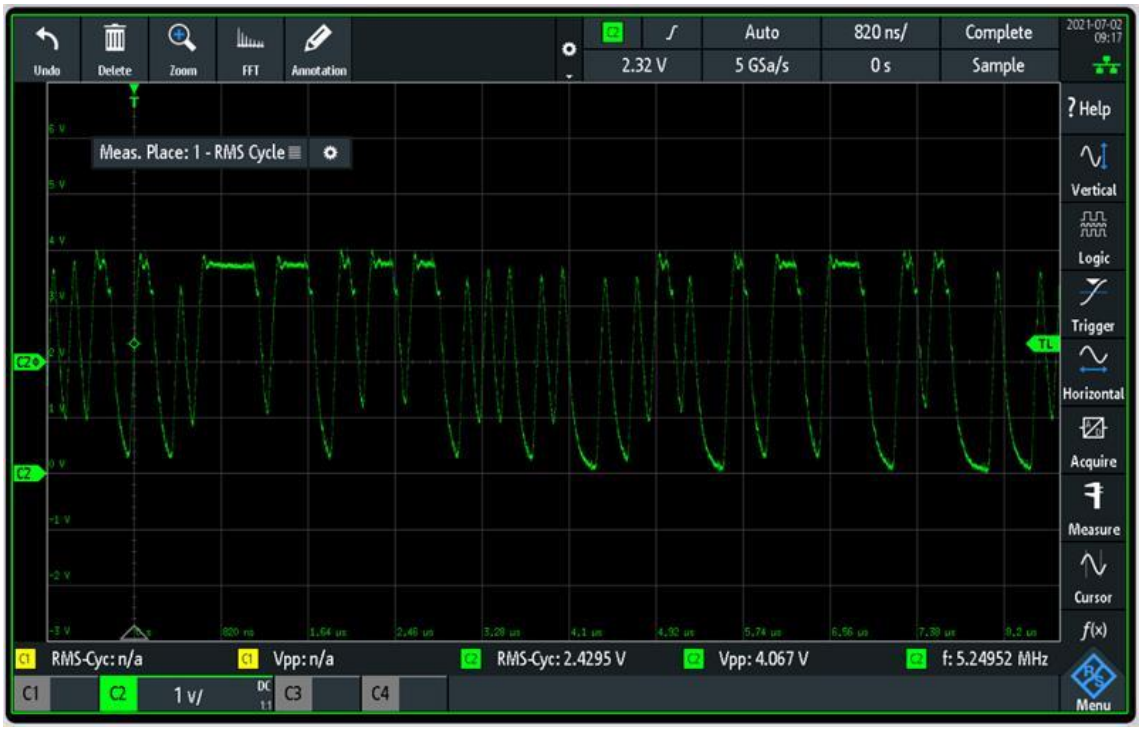


(b)

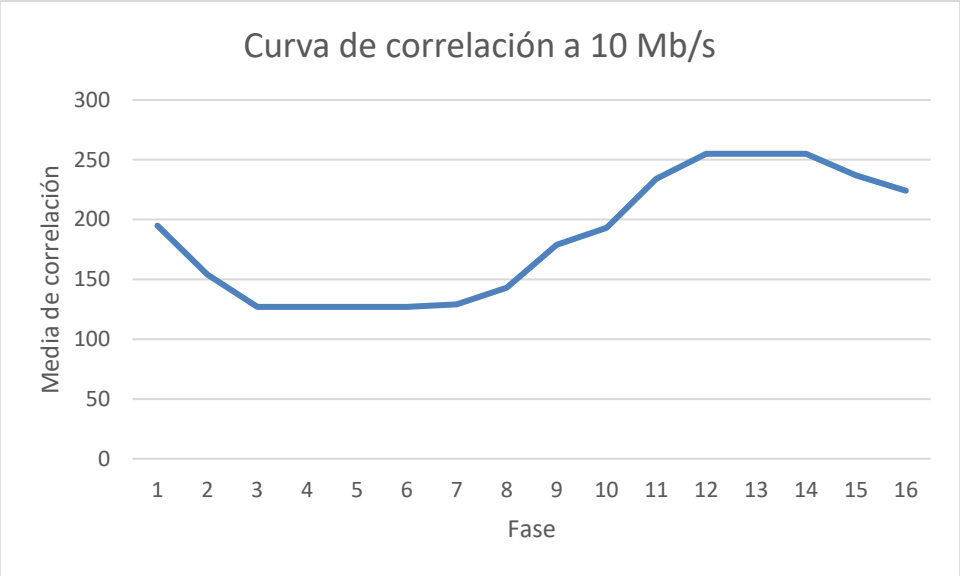
Figura 4.7 Calibración a 7.5 Mb/s. (a) secuencia de calibración capturada en el osciloscopio. (b) curva de correlación.

Se consigue un buen BER de 1.2188×10^{-7}, transmisiones a esta frecuencia serían viables.

Para 10 Mb/s los resultados se muestran en Figura 4.8. Se observa un deterioro de la señal. La curva de correlación se estrecha de forma significativa. La fase óptima es la 14, que coincide con la fase central de entre todas las que superan el umbral, que es 191.



(a)



(b)

Figura 4.8 Calibración a 10 Mb/s. (a) secuencia de calibración capturada en el osciloscopio. (b) curva de correlación.

Se obtiene un BER de 0.011, a esta frecuencia la perdida de datos es significativa.

Las medidas tomadas son coherentes con la respuesta esperada del sistema. A baja frecuencia la señal recibida es buena y muchas de las fases consiguen una correlación máxima. A 2.5 Mb/s y 5 Mb/s debería dejarse el sistema en marcha durante horas para comenzar a ver algún error.

A medida que aumenta la frecuencia de transmisión, la señal recibida empeora, la curva de correlación se estrecha y el BER aumenta.

Capítulo 5. Conclusiones

Se ha cumplido con los objetivos planteados al inicio del proyecto. Se ha modelado un sistema en Matlab que permite transmitir una trama de datos modulados en OOK-NRZ a distintas frecuencias. Permite recibir la secuencia transmitida y seleccionar el momento óptimo para capturar los bits recibidos. Y además puede realizar el cálculo del BER.

Este modelo se ha trasladado a SystemVerilog donde se ha implementado el hardware y se ha verificado tanto con ModelSim, como montado en la placa de desarrollo con SignalTap. Durante el diseño se han descompuesto las diversas funcionalidades que debe cumplir el sistema en varios módulos para simplificar el diseño y poder localizar los errores con más facilidad durante la depuración. Separando el problema a resolver en partes más manejables, ha sido posible ir construyendo pieza a pieza el sistema final.

Se ha desarrollado una interfaz entre la placa de desarrollo donde se monta el sistema y el PC a través de Matlab y una red TCP/IP. Esta interfaz permite la configuración de ciertos parámetros del sistema, así como la lectura de registros con datos relevantes del proceso de calibración y cálculo del BER. Esta parte ha sido además conveniente para poder depurar el sistema completo en la FPGA desde una ubicación remota.

Finalmente, se ha incluido el sistema diseñado para caracterizar una transmisión a través de LEDs, en el sistema montado en el laboratorio para transmitir y recibir señales con LEDs de iluminación. Se ha testeado tanto el proceso de calibración como el de cálculo del BER, y los resultados obtenidos han seguido la línea de lo que se esperaba obtener en las frecuencias configuradas. A bajas frecuencias la curva de correlación es muy ancha y el BER muy bajo. A medida que aumenta la frecuencia de transmisión de datos, la curva de correlación se estrecha y el BER aumenta.

Capítulo 6. Actividades a desarrollar en el futuro

Muchas de las ampliaciones y mejoras a introducir en el sistema se reconocieron durante las pruebas de laboratorio. La mayoría conllevan la conversión de parámetros de configuración, a puertos de configuración para poder cambiar la configuración del sistema desde la interfaz en Matlab sin tener que recompilar. La única desventaja es que esto conlleva fijar el tamaño de los puertos al máximo que se prevé que utilizará el sistema en lugar de ser adaptable. Los parámetros a modificar son:

- N_C: Con el número de veces que se repite la secuencia de calibración por fase.
- SC: Con el número de cuentas por secuencia de calibración por si el retardo es mayor de lo esperado.
- ERROR_LIMIT: Para modificar el número de errores que se contabiliza en la fase del cálculo del BER.
- I_BER: Para modificar la semilla inicial del LFSR con la secuencia a transmitir.
- PH: Para poder cambiar el número de fases en las que se captura cada bit recibido. Esto implica configurar la memoria con el máximo de fases que se prevé que utilizará el sistema.

Una de las mejoras más críticas del sistema es la utilización de la IP PLL Reconfig IP Core para generar el PLL. Este PLL, permite además de modificar la fase dinámicamente, reconfigurar los relojes de salida. Esto daría flexibilidad completa en cuanto a las frecuencias seleccionables por el sistema en lugar de solo poder seleccionar entre 4. También evitaría multiplexar el reloj.

Otra ampliación interesante sería poder acceder a ciertos registros internos del sistema adicionales, como los registros con el valor de la correlación media en cada fase.

Capítulo 7. Referencias

[1] N. Chi, LED-Based Visible Light Communications, Springer.

[2] Z. Wang, Visible Light Communications, Wiley.

[3] P. A. Hoeher, Visible Light Communications theoretical and practical foundations, Hnaser.

[4] C. M. y y a. Zambano, LED BASED VISIBLE LIGHT COMMUNICATION: TECHNOLOGY, APPLICATIONS AND CHALLENGES – A SURVEY.

[5] Terasic, «DE10-Standard https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=167&No=1081&PartNo=2,» [En línea].

[6] F. Vahid, Digital Design, Wiley.

[7] «MathWorks Help Center,» https://es.mathworks.com/help/comm/ref/comm.pnsequence-system-object.html. [En línea].

[8] V. Torres, «Introducción al pocesado digital de la señal».

[9] T. W. Cusick, «Linear Feedback Shift Register».

[10] P. Schaumont, «Lecture 6: A Random Number Generator».

[11] I. Altera, «an661 Implementing Fractional PLL Reconfiguration with Altera PLL and Altera PLL Reconfig IP Cores».

[12] J. M. M. Ferrer, «Diseño y verificación de periféricos descritos en verilog para platdorm designer».

[13] Terasic, «DE10-Standard User Manual».

[14] Y. S. 1, «TCP Server - Client implementation in C https://www.geeksforgeeks.org/tcp-server-client-implementation-in-c/,» [En línea].

[15] M. H. Center, «Comunicación TCP/IP https://es.mathworks.com/help/matlab/tcpip-communication.html,» [En línea].

[16] L. -Engin, «https://www.mouser.es/ProductDetail/LED-Engin/LZ4-40CW08-0065?qs=sGAEpiMZZMu4Prknbu83y4PIPQP9w0HVLr9SE%2Fky0xI%3D,» [En línea].