Trabajo Fin de Grado
Ingeniería de las Tecnologías Industriales

Diseño de un sensor solar asíncrono para el control de la actitud en navegación espacial

Autora: Julia Sánchez Sánchez

Tutor y publicador: Antonio Javier Gallego Len

Tutor externo: Juan Antonio Leñero Bardallo

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2022

IMSE-cnm Instituto de Microelectrónica de Sevilla

CSIC Consejo Superior de Investigaciones Científicas

Trabajo Fin de Grado

Ingeniería de las Tecnologías Industriales

Diseño de un sensor solar asíncrono para el control de la actitud en navegación espacial

Autora:

Julia Sánchez Sánchez

Tutor y publicador:

Antonio Javier Gallego Len

Profesor Sustituto Interino

Tutor externo:

Juan Antonio Leñero Bardallo

Profesor Titular Facultad de Física US

Dpto. Ingeniería de Sistemas y Automática

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2022

Trabajo Fin de Grado: Diseño de un sensor solar asíncrono para el control de la actitud en navegación espacial

Autora: Julia Sánchez Sánchez
Tutores: Antonio Javier Gallego Len, Juan Antonio Leñero Bardallo

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

A todo aquel que creyó en mí

Agradecimientos

Cuando eres un niño te ves haciendo cosas grandiosas y llegando a lugares impensados. Todos los logros del mundo parecen pocos y eres el protagonista de cada uno de ellos. Así, yo siempre había pensado en todas las cosas que quería hacer y conseguir. Sin embargo, al ir creciendo, me fui dando cuenta de que llegar a los sitios es mucho más el camino que la meta. Y en ese camino, lo más importante es la compañía. Tengo claro que estoy donde estoy por aquellos que me han apoyado en cada paso.

Así que gracias, papá, mamá y Raúl, por creer en mí desde que salí el primer día de casa para ir al colegio hace ya dieciocho años, hasta el día de hoy, sin una sola duda de si lo conseguiría.
Gracias a todos aquellos que se han ido incorporando en las diferentes paradas del recorrido, sin olvidar a los que tuvieron que irse pero no por ello han aportado menos.
A Andrea, que ha aguantado todas y cada una de mis caídas. Has sido tan parte de este trabajo, y mis años universitarios, cómo yo misma. A Cristina, que ha crecido conmigo en todo esto. A mi grupo de amigas de siempre, que son las mejores supporters que podría haber imaginado.
A los amigos que me ha dado la Escuela de Ingenieros. Sabéis quienes sois y me siento muy afortunada de llevarme mucho más que educación de este sitio.
A mi familia de la Erasmus de la T-16 que llegó para no irse, que han sido una parte fundamental de esta etapa de mi vida que ha sido la universitaria.
A Víctor, y su familia, que me han acompañado en estos dos últimos años de camino y me han dado mucha más fuerza en ella de lo que podrían imaginarse.
A todos; este viaje de mi vida académica que siento de alguna forma que acaba aquí, habría sido imposible sin vosotros. Gracias.

Por último, no podía no mencionar a las personas que, con su apoyo, han colaborado en la última recta de mi trayecto, que ha sido la elaboración de este trabajo. He tenido la suerte de poder llevarlo a cabo en el Instituto de Microelectrónica de Sevilla, y desde luego he aprendido mucho más de lo aquí queda reflejado. Agradecer a Juan Antonio Leñero por ofrecerme esta oportunidad y confiar en mí. Y mencionar en particular a Rubén, que me ha cedido su tiempo desinteresadamente y ha sido parte más que esencial desde el primer dia. Nunca olvidaré que cuando lo necesité, me tendiste la primera mano de ayuda en este mundo, y espero poder algún día hacer lo mismo por alguien. Gracias igualmente a Rafa, por el apoyo y la comprensión.
Por último, gracias a mi querida Banda del Patio por estar ahí cada día de estos últimos cuatro meses. Habéis sido un respiro diario en todo esto.

Resumen

En este Trabajo Fin de Grado se ha diseñado el píxel base de un sensor solar asíncrono. Los sensores solares son sistemas empleados para la determinación de la posición relativa del sol, siendo en concreto una de sus aplicaciones el control de la actitud en sistemas de navegación espacial (a ello se dirige el diseño realizado en este estudio). Estos dispositivos han visto su demanda disparada recientemente debido a la liberación del uso del espacio y el despliegue masivo de satélites, aumentando paralelamente la exigencia en sus especificaciones de diseño y la búsqueda de nuevas arquitecturas que sean más eficientes para el ámbito espacial. Recientemente se han propuesto arquitecturas novedosas de sensores solares digitales que permiten mayor velocidad, simplicidad de operación y menor consumo de energía. La finalidad de el diseño realizado en este Trabajo Fin de Grado era lograr la integración de dos arquitecturas de sensores asíncronos existentes, cada una de ella con ventajas específicas (sensores de tipo Octopus y sensores de tipo Continuo), de forma que sea posible seleccionar el modo de funcionamiento según las especificaciones de la situación a controlar.

Para alcanzar este objetivo, se combinaron los diseños de los sensores previos y se incorporaron nuevas señales de control e internas que, combinadas mediante circuitos lógicos y con *enables* de ciertos elementos, permitiesen esta conmutación de método de funcionamiento. Para comprobar que en efecto el diseño planteado cumplía satisfactoriamente con las necesidades establecidas, se realizó el diseño y se validó mediante la simulación de la respuesta de una muestra de un tamaño reducido de la matriz de píxeles (ante diferentes escenarios de iluminación). Los resultados obtenidos, que se presentan en este trabajo, verificaron que el sistema proyectado respondía convenientemente según las especificaciones aportadas por el grupo de trabajo donde se realizó el TFG. También se sometió al sistema a análisis de *corners* y análisis estadísticos para asegurar su fiabilidad. Todo lo anterior se llevó a cabo en el programa *Cadence*, lo cual permitió a la autora la familiarización con el flujo de diseño de circuitos integrados con una herramienta profesional.

Este Trabajo de Fin de Grado se realizó en el Instituto de Microelectrónica de Sevilla gracias a una Beca JAE de introducción a la investigación, y los resultados obtenidos serán empleados para completar el diseño de un sensor solar futuro por el grupo de investigación (grupo TIC-179 de la US) en el que la autora trabajó.

La memoria de este Trabajo se organizó en seis capítulos: en el primero, se expone brevemente una introducción y motivación del estudio a realizar. En el segundo, se lleva a cabo una revisión del estado del arte, planteándose las arquitecturas de los píxeles en los que se basó este diseño y las ventajas del desarrollo de este. A continuación, en el tercer capítulo encontramos detallada la estructura del píxel desarrollado y sus componentes. En el cuarto capítulo se analizan los circuitos periféricos de lectura de la matriz necesarios para el correcto funcionamiento del sensor. Posteriormente, en el el capítulo cinco encontramos las diferentes simulaciones y análisis que se llevaron a cabo para verificar la funcionalidad del diseño. Por último, en el sexto capítulo se plantean las conclusiones extraídas y las líneas de trabajo futuro.

Abstract

A pixel for an asynchronous solar sun sensor has been designed in this degree final project. Solar sun sensors are systems used to determine the relative position of the Sun, and one of their applications is the control of the attitude on space navigation systems (this work is aimed for this use). These devices demand has sky-rocketed reciently due to the liberalization of space use and the massive unfold of satellites, and consequently, the requirements on their design specifications have rised, as well as the search for new architectures more efficient for space field. Reciently novel architectures of digital solar sun sensors have been proposed, and those allow a bigger speed, simplicity of operation and a lower energy consumption. The objective of the design developed in this degree final project is to integrate two existing architectures of asynchronous solar sun sensors, each one of them presenting specific advantages (Octopus sensors and Continuous sensors), in a way that permits us selecting the operating mode depending on the specifications of the situation we have to control.

To reach this objective, previously developed sensor designs were combined, incorporating new internal and control signals that allowed the conmutation between operating modes when combined using logic circuits and enable signals for some components. To test that the suggested design achieved satisfactorily the stablished demands, the design was implemented and validated through a simulation of the response of a sample of a smaller size of the pixel matrix to different lighting scenarios. The obtained results, presented in this work, verified that the projected system answers conveniently. In addition, the system was subjected to a Corners Analysis as well as statistical analysis. Everything was carried out with Cadence software and tools, what allowed the author to familiarize with integrated circuits design flow in a professional tool.

This work was conducted in the Seville Institute of Microelectronics thanks to a JAE Intro Scholarships of CSIC, and the results obtained will be used to complete the design of future solar sun sensors in the research group the author worked at (group TIC-179 of University of Seville).

The report of this work is organized in six chapters: in the first one, a introduction and motivation of this study is briefly exposed. In the second one, a revision of the state of the art is carried out, presenting the pixel architectures on which this designed is based and the advantages of it. Following, in the third chapter we find detailed the designed pixel structure and its components. In chapter four the periferal circuits used to read the matrix are analyzed. Afterwards, in the fifth chapter we find the different simulations and analysis that were carried out in order to verify the funtionality of the design. Finally, in chapter six the conclusions obtained from this study are presented and the possible future work lines are discussed.

Índice Abreviado

Índice

Notación

AER	*Adress Event Representation*
FD	*Focal Distance* - Distancia Focal
FOV	*Field of View* - Campo de Visión
FR	*Free Running*
ROI	*Region of Interest* - Región de Interés
TFS	*Time to First Spike*
UMC	*United Microelectronics Corporation*
US	Universidad de Sevilla

1 Introducción y motivación

En las últimas décadas, la industria aeroespacial ha experimentado un crecimiento exponencial, debido a múltiples factores. En primer lugar, la reducción de costes de fabricación y puesta en órbita de sistemas de navegación ha permitido que la investigación en la materia se viera potenciada. Así, se ha acelerado el progreso en el desarrollo de satélites de reducidas dimensiones (micro, nano y pico-satélites) [1]. En segundo lugar, y no por ello menos importante, en los últimos años ha comenzado la denominada *carrera espacial privada*: tanto el sector privado como el público han visto en la exploración espacial múltiples oportunidades de negocio y crecimiento, desde el turismo espacial de transporte de personas (que empieza a plantearse en nuestros días) hasta la explotación de recursos naturales extraterrestres. Así, encontramos involucrados en esta expansión tanto a magnates de la tecnología como a empresas y centros de investigación de un tamaño más discreto. Veáse por ejemplo la Universidad de Oslo, que cuenta con su propio programa espacial y lanza anualmente una sonda a la ionosfera con sistemas ideados por alumnos. Todo esto ha causado que aumente el interés y la dedicación a la mejora de la tecnología existente.

Los sistemas de determinación de actitud son subsistemas cruciales en toda navegación espacial, ya sean satélites, cohetes de exploración, drones o sistemas de navegación espacial. El objetivo de éstos es determinar la orientación del objeto espacial respecto de un sistema de referencia inercial. Este control de actitud es uno de los puntos en los que convergen actualmente el estudio de la teoría de control y el sector aeroespacial. Para poder llevar a cabo este cometido son necesarios, entre otros elementos, sensores que capten la información del medio para su correcto procesamiento. Distinguimos en éstos, principalmente, dos clases: absolutos y relativos. Los sensores absolutos son aquellos que determinan la posición u orientación de campos, objetos u otros fenómenos externos al objeto espacial en el instante actual. Entre ellos se incluyen los sensores solares, sensores de horizonte terrestre, magnetómetros, etc. Los sensores relativos tienen como objetivo producir una señal de salida que prediga el ritmo de cambio de la actitud del objeto espacial. Así, basándose en una actitud inicial conocida o en información externa, emplean estos ritmos de cambio para calcular la actitud actual. Incluimos en ellos los acelerómetros, giróscopos, etc.

Este Trabajo Fin de Grado se centrará en el estudio de los sensores solares como herramientas para la determinación de la actitud de objetos espaciales. [2] Para esta utilidad, encontramos unas necesidades por satisfacer más exigentes que las de un sensor solar de uso comercial como pueden ser aquellos dedicados a la orientación de dispositivos de recolección de energía fotovoltaica, al control de la iluminación... En concreto, se requiere un tiempo de respuesta bajo, un reducido consumo de energía, poco peso, volumen limitado, tolerancia a radiación espacial y el cumplimiento de las especificaciones de funcionamiento de la aplicación como pueden ser la precisión, resolución y el campo de visión.

Habitualmente, los sensores solares empleados con el objetivo de la determinación de la actitud están compuestos por una matriz de píxeles cubierta por una lámina de tipo lente estenopeica (con un orificio de dimensiones muy reducidas) de un material opaco, que causará que dependiendo de la orientación del Sol se ilumine una determinada región de la matriz. Periódicamente, se extraerá la información sobre el nivel de iluminación de cada píxel. Esta implementación, que se explicará detalladamente más adelante en este documento, presenta un inconveniente: al leerse todos los píxeles de la matriz se genera un gran flujo de datos con información redundante, puesto que gran parte de la matriz de píxeles no se halla iluminada. Esto genera una ralentización de la operación y un consumo de energía y ancho de banda relativamente elevado, lo

cual va en contra de los requerimientos especificados anteriormente para un sensor incorporado a un objeto de navegación espacial.

Con el propósito de solventar estas desventajas y cumplir con las exigencias planteadas, el grupo de investigación de Microelectrónica Analógica y de Señal Mixta de la US (grupo TIC-179) ha diseñado varios prototipos [3, 4, 5, 6] de sensores solares asíncronos, que lean sólo la información de los píxeles iluminados y sean capaces con estos datos de calcular la posición del Sol, permitiendo su diseño reducir el flujo de datos al exterior y el consumo, y simplificando los protocolos de comunicación e integración con otros posibles elementos del satélite al que se incorporen. La tecnología empleada para su fabricación fue UMC180nm. Así, el principal objetivo de este Trabajo Fin de Grado es proponer una arquitectura de píxel que reuna las funcionalidades de varios de estos prototipos diseñados y contenga la circuitería necesaria para poder cambiar el funcionamiento de los píxeles entre uno u otro modo.

A continuación, encontramos el resto de este trabajo divididos en cinco capítulos.

- En el Capítulo 2 se presentan los tipos de sensores solares desarrollados hasta el momento para esta aplicación y se expone el modo de operación de cada uno de ellos. Una vez introducida la materia de interés, se plantean las ventajas resultantes del diseño propuesto en este estudio.
- En el Capítulo 3 encontramos la arquitectura del píxel del diseño alcanzado y una explicación de ésta, sus componentes y sus funcionalidades.
- En el Capítulo 4 se exponen las estructuras de las periferias que junto con la matriz de píxeles del diseño planteado en el capítulo tres, compondrán el sensor solar.
- En el Capítulo 5 se presentan los resultados de la simulación de este sistema en el entorno *Cadence*.
- Por último, en el Capítulo 6 encontramos las conclusiones obtenidas y un planteamiento de las posibles líneas futuras de trabajo. Se incluye para finalizar una comparativa del prediseño con algunos sensores existentes.

2 Revisión del estado del arte

En este capítulo se abordará la contextualización del trabajo realizado y la presentación de los principales conceptos de la física e ingeniería en los que se basará el desarrollo del sistema propuesto. Así, se revisan las clases de sensores solares y se profundiza en los modelos desarrollados por el grupo de investigación TIC-179 de la US.

Como ya se introdujo en el Capítulo 1 de este Trabajo Fin de Grado, hoy en día los sensores solares son frecuentemente empleados en la industria aeroespacial para la determinación de la actitud [2]. Éstos son dispositivos que determinan la posición relativa del sol referida a su centroide, determinando la latitud, θ, y azimut, ϕ, de éste respecto del objeto espacial. Vemos una representación de estas magnitudes en la Figura 2.1.

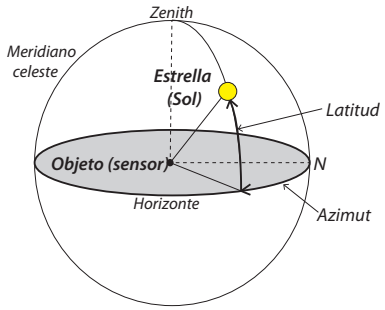

Figura 2.1 Esquema representativo de las coordenadas angulares que determinan la posición relativa del Sol.

2.1 Clasificación de los sensores solares

2.1.1 Sensores solares analógicos y digitales

Los sensores solares se dividen fundamentalmente en dos grandes grupos, según la arquitectura que emplean para la detección de la orientación: analógicos y digitales. En este trabajo se presentan cada uno de estos tipos particularizados para la función de detección de la orientación del Sol en navegación espacial.

Los sensores analógicos [7, 8, 9] no necesitan hardware adicional; están compuestos por dos fotodiodos fabricados en el mismo sustrato de silicio cristalino y colocados ortogonalmente entre sí. El ángulo de incidencia de la luz es mesurable mediante el cociente de las corrientes generadas en cada fotodiodo. Así, presentan como ventaja su simplicidad y rapidez de operación, pero son muy sensibles a la iluminación del ambiente (efecto albedo), son propensos a sufrir *mismatch* y tienen poca precisión. El efecto albedo consiste en la incidencia de la radiación que proviene de el reflejo de la radiación del Sol u otros cuerpos estelares en cuerpos cercanos al sensor [10, 11, 12]. Estos datos no son los que pretendemos analizar y pueden falsear

el resultado. Además, dado que las corrientes generadas en los fotodiodos son débiles, se requerirán varias etapas amplificadoras.

Los sensores digitales [5, 13, 14, 15] están compuestos por una matriz de píxeles (cada píxel contendrá un fotodiodo y su circuitería asociada) que llevará acabo el sensado, y una periferia que llevará a cabo la lectura y el control de la matriz y el procesamiento/comunicación al exterior de la información. Suelen emplearse píxeles *Active Pixel Sensor* (APS) [16]. Tradicionalmente, la matriz es leída de forma secuencial y síncrona por la periferia, dando como salida el nivel de iluminación de cada píxel de la matriz. Posteriormente un algoritmo, implementado en la periferia o en un dispositivo externo, procesa esta información para calcular el centroide de la Región de Interés (a la que de ahora en adelante denominaremos *Region of Interest*(ROI).

Al contrario de los analógicos, los sensores digitales son fiables, precisos, y robustos frente a las perturbaciones de luz del entorno. No obstante, presentan inconvenientes, como el alto consumo derivado de la lectura periódica de toda la matriz que requieren los sensores digitales convencionales. Además, el tiempo de integración (tiempo entre la incidencia de la luz y la generación de la corriente en el fotodiodo) ha de ser predefinido, y el tiempo de exposición ha de ajustarse a las condiciones de iluminación. Por otra parte, necesitarán de periféricos externos (FPGAs o microcontroladores) para el cálculo del centroide.

La implementación física del sistema de los sensores digitales que se lleva a cabo para poder hallar estas magnitudes se basa en la Óptica Estenopeica (*Pinhole Optics*) [4]. Se expondrá esta implementación en detalle puesto que es la empleada en los modelos de sensores que se estudian en este trabajo. Consiste en una matriz de píxeles de $M \times N$ (cuyo tamaño dependerá del campo de visión necesario) sobre la que se coloca una lámina ultra-fina de un material opaco que tiene en su centro un agujero de dimensiones muy reducidas. La luz penetrará a través de este agujero microscópico, iluminando una región de la matriz, que será la ROI. Calculando las coordenadas en la matriz de píxeles del centroide de la ROI podremos determinar la posición del sol, tal y cómo se muestra en la Figura 2.2.

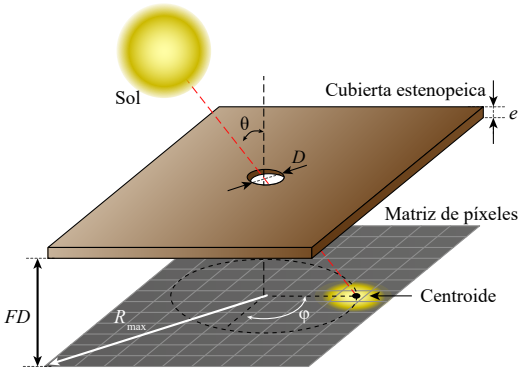

Figura 2.2 Esquema representativo de la implementación física del sistema.

Una vez hayamos determinado el centroide de la ROI (de coordenadas (x, y)), la posición del Sol quedaría definida por las ecuaciones 2.1 y 2.2 [5, 17].

$$\theta = \arctan\left(\frac{\sqrt{W^2 \cdot (x-x_0)^2 + L^2 \cdot (y-y_0)^2}}{FD}\right) = \arctan\left(\frac{R}{FD}\right) \tag{2.1}$$

$$\phi = \arctan\left(\frac{L \cdot (y-y_0)}{W \cdot (x-x_0)}\right) \tag{2.2}$$

donde W y L son el ancho y el largo de un píxel, respectivamente, (x_0, y_0) son las coordenadas del centroide de la ROI para $\theta = 0^\circ$ y FD es la distancia focal (distancia entre la matriz de píxeles y la lámina opaca). Como se mencionó anteriormente, el tamaño de la matriz dependerá del campo de visión (FOV, *Field of View*) necesario. Este campo se define como el el ángulo abarcable a través del cual el sensor puede detectar la radiacción electromagnética, y podemos calcularlo a partir del radio máximo (distancia de una de las

esquinas al centro de la matriz) y la distancia focal:

$$R_{max} = \frac{1}{2} \cdot \sqrt{(W \cdot M)^2 + (L \cdot N)^2} \tag{2.3}$$

$$\theta_{max} = \arctan\left(\frac{R_{max}}{FD}\right) \tag{2.4}$$

$$FOV = 2 \cdot \theta_{max} = 2 \cdot \arctan\left(\frac{R_{max}}{FD}\right) \tag{2.5}$$

2.1.2 Sensores solares asíncronos

Como solución al alto consumo que presentan los sensores digitales (debido a la gran cantidad de píxeles a leer cada vez que se quiera determinar la actitud) se plantean los sensores solares asíncronos [5, 4, 6]. El desarrollo de estos es iniciado con el objetivo de emular, mediante un dispositivo electrónico, la transmisión de la información asíncrona mediante impulsos nerviosos que ocurre en el nervio óptico [18, 19].

Desde que en 1970 se propusiera el primer modelo de píxel intentando imitar el comportamiento de la retina humana [20] se han hecho grandes avances que han permitido llegar a modelos funcionales, eficientes e integrados (el píxel de Fukushima tenía una implementación con componentes discretos). Durante esta evolución, cabe destacar como hito crucial la invención del protocolo de comunicación de representación de dirección de eventos (*Adress Event Representation, AER*) [21, 22, 23] cuyo objetivo es gestionar la transmisión de información mediante peticiones realizadas por cada unidad receptora cada vez que se detecta un nuevo estímulo. Se explicará en detalle este protocolo más adelante en este capítulo.

Dentro de estos modelos de sistemas bio-inspirados, el grupo de investigación TIC-179 de la US ha desarrollado en particular dos modelos de sensores, que se exponen a continuación, y que son en los que se profundizará en este Trabajo Fin de Grado [4, 5].

2.2 Sensor Octopus

Los sensores Octopus son aquellos cuyo principio de funcionamiento consiste en codificar en la frecuencia o ancho de pulso de la señal de salida los niveles de iluminación de cada píxel. La primera propuesta en esta línea de investigación la realizó Culurciello [24] en 2003, planteando un píxel compuesto por un fotodiodo integrador de carga y un comparador que activase su salida cuando la tensión acumulada en el fotodiodo superase un nivel umbral. Así, se generaría una salida con forma de tren de pulsos proporcional al nivel de iluminación. Para conseguir esto, se envía una señal de reset al fotodiodo cada vez que el píxel es leído por la periferia.

J.A. Leñero et al. [3] presenta un píxel con capacidad de autoreseteo que genere una señal de salida (*spike*) cuya frecuencia sea proporcional al nivel de iluminación. En la Figura 2.3 encontramos un esquema simplificado de un píxel Octopus y su modo de operación.

La capacidad C que aparece en paralelo al fotodiodo $D1$ representa la capcidad equivalente en el cátodo del fotodiodo, I_{ph} la fotocorriente generada y V_c es por tanto el voltaje acumulado por el fotodiodo durante la integración de la carga. Como podemos ver con más claridad en la gráfica que representa V_c en función del tiempo, la tensión del fotodiodo se precarga inicialmente a la tensión de alimentación V_{DD}, y a continuación comienza a descargarse con una pendiente proporcional a la iluminación hasta que llega al valor umbral V_{ref}, momento en el que genera un pulso que realizará la petición y a la vez reseteará el fotodiodo precargándolo de nuevo mediante el transistor $M1$.

Debido a la naturaleza del fotodiodo, éste puede actuar tanto como elemento que consume potencia o como elemento generador de ésta, dependiendo de cómo se polarice. Así, un fotodiodo puede encontrarse en los regímenes de operación mostrados en la Figura 2.4 [25].

En el píxel Octopus explicado anteriormente, el fotodiodo está polarizado en inversa de forma que está consumiendo corriente y está en su zona de funcionamiento conocida como fotodetectora. Si polarizamos

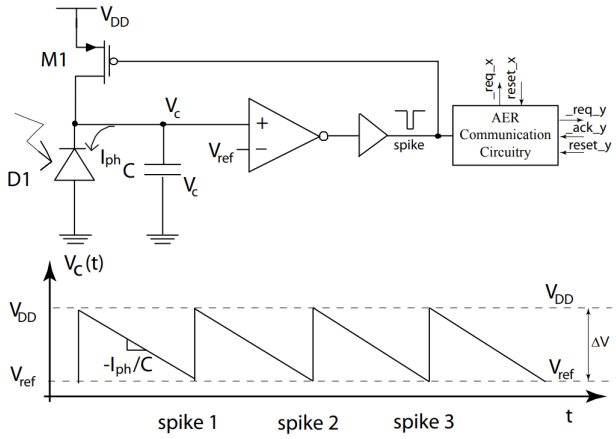

Figura 2.3 Diagrama ilustrativo de la implementación de un píxel Octopus que genere pulsos de frecuencia proporcional al nivel de iluminación.

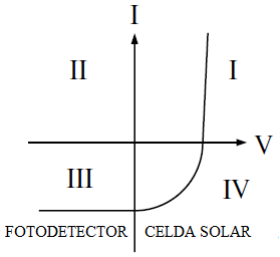

Figura 2.4 Representación I-V de los diferentes regímenes en los que funciona un fotodiodo.

este fotodiodo en directa y hacemos que trabaje en su zona de funcionamiento como celda solar, podríamos lograr que no consumiera corriente, lo cual para aplicaciones en las que se busca bajo consumo es lo óptimo. Esta implementación sería la que se muestra en la Figura 2.5 y es la que se empleará en este Trabajo Fin de Grado.

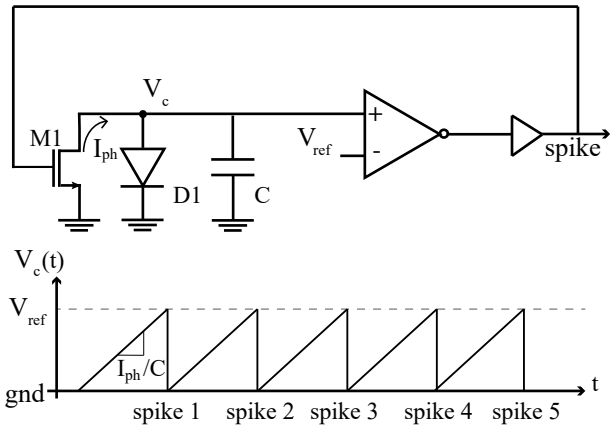

Figura 2.5 Diagrama ilustrativo de la polarización de un fotodiodo como celda solar dentro del píxel Octopus.

2.2.1 Protocolo de comunicación AER

Para explicar el funcionamiento de un píxel Octopus es vital la presentación del protocolo AER, *Adress Event Representation*, ya mencionado anteriormente como pilar de los modelos asíncronos de píxeles bio-inspirados que existen a día de hoy. Mientras que en el cerebro el conexionado entre neuronas es masivo (punto a punto), el protocolo AER multiplexa en el tiempo el uso de un único bus (canal compartido) lo cual es posible dado que los sistemas electrónicos son mucho más rápidos que los biológicos. Así, estamos consiguiendo no tener que interconectar todos los píxeles (elemento equivalente en este caso a las neuronas) con el receptor, lo cual nos permite emular la comunicación interneuronal cerebral superando las limitaciones de rutado que presentaría generar un modelo morfológicamente similar en un circuito electrónico.

La principal dificultad a solventar a la hora de crear un sistema asíncrono es la comunicación: cómo distinguir qué píxel de la matriz ha de ser leído, puesto que los píxeles generan información de forma autónoma e independiente. Es como solución a esta cuestión que surge el protocolo AER [21, 22, 23], cuyo funcionamiento está basado en conectar los elementos emisores (cada píxel en nuestro caso) con el receptor (el elemento externo de procesamiento de información) mediante un proceso de comunicación que involucra una petición de pedida y aceptación de ésta previas a la lectura de datos. Estos dos elementos (emisor y receptor) serán las entidades principales del protocolo. En la Figura 2.6 se ilustra la comunicación entre ellos, que tiene lugar en los siguientes pasos, marcados en el cronograma de dicha figura:

1. El emisor detecta un evento y emite una petición, activando la línea *REQ*.
   El receptor recibe la petición y carga la información en el bus de datos. El proceso de recepción de la petición dependerá de cuántas se estén realizando simultáneamente y cómo se haya decidido priorizarlas. Más adelante en este Trabajo Fin de Grado se expondrá cómo se arbitran en nuestro diseño.
2. El receptor activa la señal *ACK* cuando la información ha sido cargada.
3. Al recibir el *ACK*, el emisor desactiva REQ.
4. Cuando se desactiva la petición, el emisor deja de transmitir la información al bus de datos.

Figura 2.6 Representación de los elementos y las líneas que los comunican en el protocolo AER. Representación de un intercambio de información entre ellos.

2.2.2 Modos de funcionamiento

El sensor Octopus [5] se diseñó con el objetivo de obtener como salida de cada píxel un vector que contuviera codificado en su frecuencia la intensidad luminosa que recibía el fotodiodo, y que este vector fuese interpretado por un dispositivo externo que conociendo el nivel de iluminación de cada píxel calculase el centroide del área iluminada. Este tipo de codificación de la información es conocida como modulación PDM (*Pulse Density Modulation*, Modulación de Señal por Densidad de Impulsos) [26]. Sin embargo, se comprobó [5] que el píxel más iluminado se halla siempre muy cercano a la posición del centroide. Este píxel emite el pulso en primer lugar (mayor nivel de iluminación implica mayor velocidad de carga), por lo que recibiendo un solo evento, podría determinarse con relativa exactitud la posición del Sol. Así, se concluyó que para un mayor número de eventos procesados se reduce el error de medida pero se aumenta el tiempo de procesamiento. Como consecuencia de este hecho, se desarrollaron dos modos de funcionamiento para el Sensor Octopus, en función de si se desea priorizar la exactitud o el tiempo de respuesta.

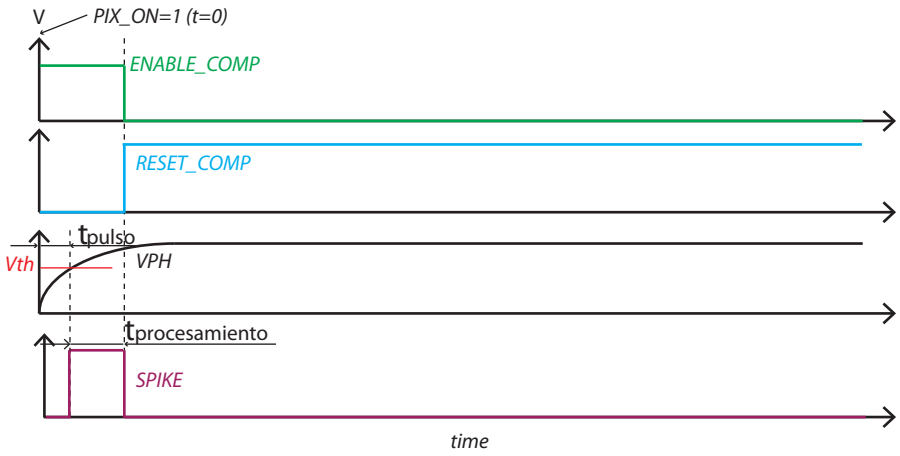

Figura 2.7 Cronograma ilustrativo de la evolución temporal de señales en el píxel para el funcionamiento en modo TFS. *Pix_on* ≡ Señal de estado de medida (activa cuando se está midiendo, inactiva cuando no), *Enable_comp* ≡ señal de activación del comparador, *Reset_comp* ≡ señal de bloqueo del comparador, *v_ph* ≡ voltaje en el ánodo del fotodiodo, *SPIKE* ≡ señal de pulso a la salida del comparador.

Modo Time-to-First-Spike

Como su propio nombre indica (tiempo hasta el primer pulso) este modo consiste en hacer que cada píxel emita un solo evento, y posteriormente se bloquee. Así, la señal de salida será un vector de tiempos que contendrá el tiempo que ha tardado cada píxel en dar el primer pulso [5]. De este vector, podremos extraer dos líneas de datos: la posición del centroide aproximada por el píxel que primero haya alcanzado la tensión umbral, o la intensidad luminosa de cada píxel codificada en el tiempo que ha tardado en emitir el pulso. Si se decidiera calcular el centroide con esta segunda opción, las ecuaciones que deberían implementarse en el dispositivo externo de procesamiento de la información serían:

$$\left.\begin{aligned} x_c &= \sum_{i=0}^{M} \frac{\sum_{j=0}^{N}(im(i,j)\cdot i)}{\sum_{j=0}^{N} im(i,j)} \\ y_c &= \sum_{i=0}^{M} \frac{\sum_{j=0}^{N}(im(i,j)\cdot j)}{\sum_{j=0}^{N} im(i,j)} \end{aligned}\right\} \tag{2.6}$$

donde $im(i,j)$ representa el nivel de iluminación del píxel de coordenadas (i,j), y (x_c,y_c) serían las coordenadas del centroide. En caso de que se optara por esta opción, el cálculo del centroide es el mismo que en el caso de tomarse todos los pulsos emitidos durante el tiempo de medida, sólo que la precisión es menor, dado que la alta demanda de peticiones simultánea causa desviaciones entre el tiempo real en el que se genera el pulso y el tiempo medido que llega al periférico externo. En el modo de funcionamiento FR, el impacto de este fenómeno se reduce ya que al tomar más de un pulso el error promedio disminuye.

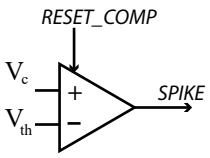

Figura 2.8 Símbolo de un comparador con *RESET*.

A nivel de diseño electrónico, conseguiremos implementar este modo colocándole al comparador un transistor de *ENABLE* y otro de *RESET* que activen el comparador al comenzar el periodo de medida y consigan bloquearlo una vez se ha emitido el primer pulso. El comparador es el bloque analógico que más consumo tiene del píxel, por tanto la incorporación de estas dos señales nos aporta otra gran ventaja que

es la reducción de este consumo cuando se halla desactivado. Cómo se coordinan estas señales para la implementación del modo TFS queda reflejado en la Figura 2.7.

Modo Free-Running

Este modo (libre ejecución, en español) es el que se describió por defecto para el sensor Octopus [5], y consistirá en recoger en la señal de salida todos los pulsos emitidos por cada píxel durante el periodo que estemos midiendo. Así, se calculará la posición del centroide con el sistema de ecuaciones 2.6, pero tal y cómo se comentó anteriormente, la precisión será mayor.

Para implementar este modo en el diseño electrónico, simplemente haremos que la señal *ENABLE* del comparador sea la misma señal que comunica que está activo el periodo de medida, y *RESET* será la negada de esta. Vemos un cronograma ilustrativo de cómo evolucionarán estas señales en la Figura 2.9

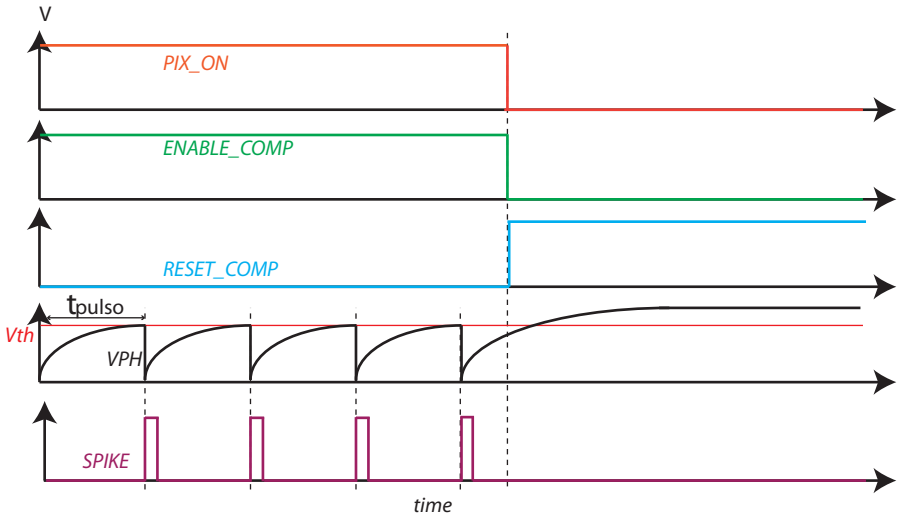

Figura 2.9 Cronograma ilustrativo de la evolución temporal de señales en el píxel para el funcionamiento en modo FR.

2.3 Sensor Continuo

Como una alternativa al sensor Octopus, que presenta cierta complejidad de operación al incorporar el protocolo AER, el grupo de investigación TIC-179 de la US propuso [4] un nuevo concepto de sensor solar asíncrono: el sensor de operación continua. La motivación que llevó a su desarrollo fue el poder calcular el centroide directamente en el chip a alta velocidad. Este sensor, como su propio nombre indica, opera de forma continua, con unos tiempos de retardo muy bajos, y calcula directamente en la propia periferia del chip la posición del centroide. Es decir, las señales del salida del sensor son las coordenadas del centroide (y no un vector con información luminosa como en el caso del sensor Octopus). Fue en este sensor en el que por primera vez se incorporó el fotodiodo como celda solar para la aplicación de estos como sensores solares.

El principio de operación del sensor Continuo consiste en no resetear el fotodiodo en ningún momento mientras que la señal de medida esté activa. Lo que logramos así, es que todos los píxeles iluminados activen sus líneas de petición. Estas líneas serán la entrada de la lógica circuital de la periferia, que será la encargada de calcular la línea que suponga la coordenada del centroide. Cómo se implementa este algoritmo será explicado con detalle en el Capítulo 4 de este Trabajo Fin de Grado.

Para ilustrar cómo evolucionará el voltaje del fotodiodo y la salida del comparador en cada modo de funcionamiento (recordemos la implementación del píxel presentada en la Figura 2.3), una vez se ha activado la señal de medida en el instante $t = 0$, se incluye la Figura 2.10. Como podemos observar, el fotodiodo en el modo Continuo sólo se descargará o bien cuando el píxel deje de estar iluminado (que es lo que se representa en este caso, donde vemos la curva de descarga del voltaje V_{ph}) o bien cuando se resetee el dispositivo desde

el exterior (esto no lo encontramos representado).

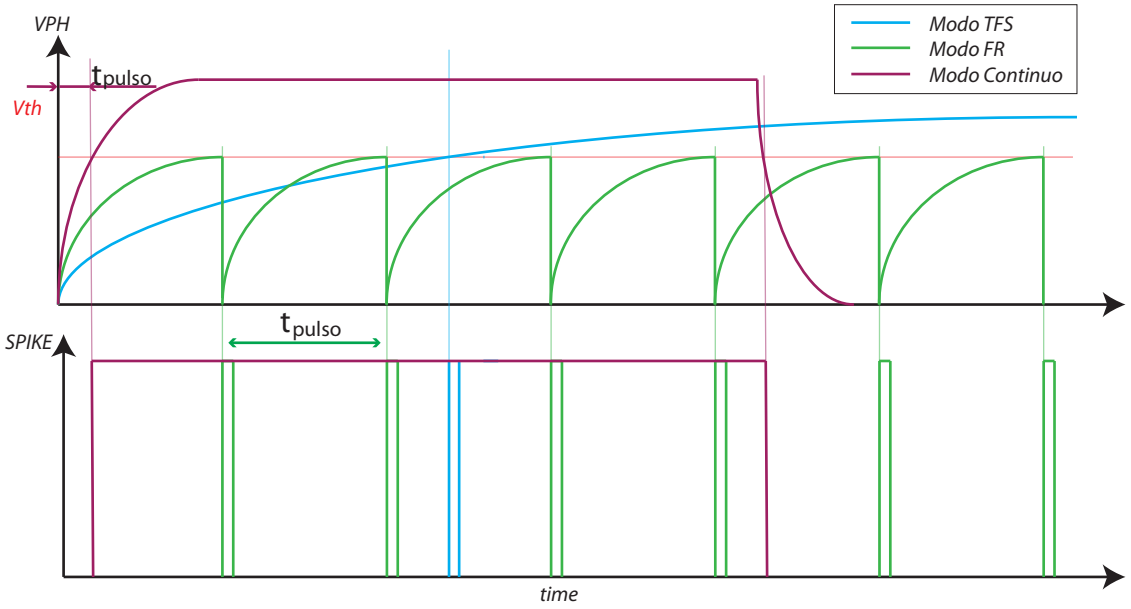

Figura 2.10 Voltaje en el fotodiodo y generación de pulsos a la salida del comparador según el modo de funcionamiento. Por simplicidad en la representación del cronograma, en esta figura se suponen intensidades luminosas distintas para cada modo de funcionamiento, de ahí que las curvas V_ph sean distintas. Para una misma iluminación, hasta el instante del primer pulso las tres curvas serían la misma.

2.4 Ventajas de la unión de las diferentes arquitecturas

Hemos planteado por tanto en las secciones anteriores de este capítulo, los modelos de sensor solar asíncrono desarrollados por el grupo de investigación: sensor solar Octopus (que admite la operación en modo TFS y modo FR) y sensor solar Continuo. El objetivo de este trabajo, como ya se introdujo en el Capítulo 1, es realizar el diseño de un píxel que pueda funcionar según los tres modos, conmutando entre uno u otro mediante señales de entrada al sensor. En esta unión encontraríamos varias ventajas, como la adaptabilidad a diferentes situaciones y la posibilidad de decidir la mejor arquitectura en función a resultados experimentales concretos más elaborados.

Este proyecto resulta interesante porque aúna las ventajas de cada modo y al permitir cambiarlos, puede resolver los inconvenientes que éstos presentan en determinadas situaciones. Estas ventajas e inconvenientes serían:

1. En el sensor Continuo encontramos una solución cerrada elegante que responde muy bien, pero que presenta fundamentalmente dos limitaciones: la resolución espacial máxima que se alcanza con el algoritmo que se implementa para el cálculo del centroide es de medio píxel, y es una solución difícil de testar pues al tener una señal de salida única que son las coordenadas del centroide no existe gran variedad de datos con los que correlacionar el resultado.

2. Ambos modos del sensor Octopus presentan como inconveniente que requieren el cálculo de centroide fuera del chip y un procesamiento posterior más complejo de los datos. Sin embargo, siempre que este cálculo se haga según el sistema de ecuaciones 2.6 tendremos en las coordenadas mayor resolución espacial. El hecho de trabajar con niveles de iluminación permite inclusive detectar varias zonas iluminadas cuando hay efecto albedo y decidir cuál será la de interés. Además, el test de este tipo de sensores tiene mucha menos dificultad. También permite elegir el diámetro de la óptica de forma adecuada viendo el tamaño de la ROI en función del diámetro del orificio. En definitiva, da un número

extenso de datos de salida que permite evaluar y testar situaciones inesperadas.

3. Dentro del sensor Octopus, el modo TFS es una variante del modo de libre ejecución que da mejor resolución temporal comprometiendo ligeramente la exactitud de los resultados [5].

Así, teniendo un píxel que pueda trabajar con los tres modos, podremos elegir qué características se adaptan mejor a la operación para la que necesitemos determinar la actitud.

3 Estructura del píxel

En este capítulo se presentará la estructura del píxel, detallando el análisis funcional de cada uno de sus elementos.

El objetivo del diseño era lograr un píxel que integrase los tres modos de funcionamiento del sensor solar explicados en el Capítulo 2 de este Trabajo Fin de Grado: el modo Octopus con configuración TFS, el modo Octopus con configuración FR y el modo Continuo.

3.1 Esquemático del píxel

El esquemático del píxel que se propone es el mostrado en la Figura 3.1. Éste es el resultado de una composición de los píxeles propuestos en (artículos continuo y octopus) y los elementos necesarios para poder cambiar el funcionamiento dependiendo del modo. En este esquemático se muestran los componentes principales del píxel como "cajas negras", presentando simplemente sus entradas y salidas y cómo se comunican. El esquemático de la implementación de píxel completo la encontramos a continuación en la Figura 3.2. Se incluirá el análisis de cada componente del circuito en las secciones posteriores a ello dedicadas.

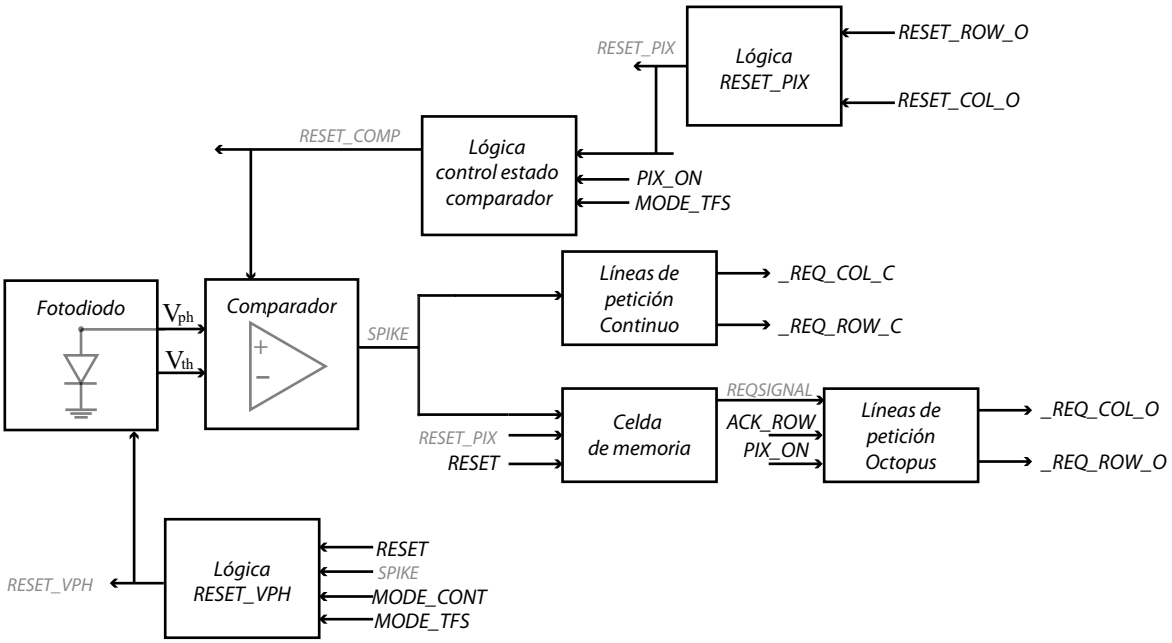

Figura 3.1 Esquemático del píxel. Aparecen en un color menos intenso y menor tamaño aquellas señales que son internas, y no entradas ni salidas.

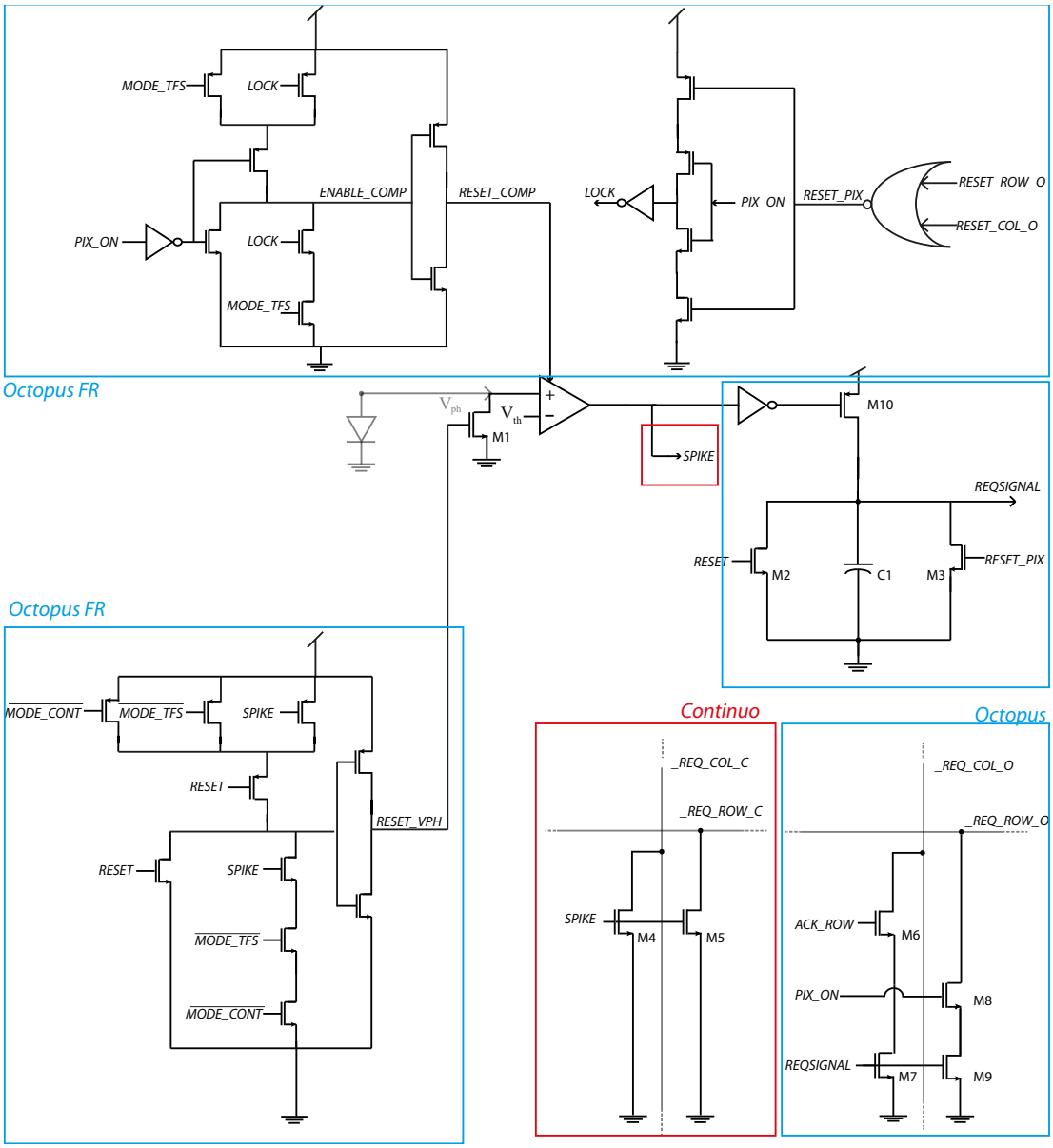

Figura 3.2 Esquemático del píxel completo. $M_{n,1} = 0{,}44/0{,}18$, $M_{n,2,3} = 4/0{,}18$, $M_{n,4,5,6,7,8,9} = 9{,}9/0{,}2$, $M_{n,10} = 0{,}25/0{,}18$. Todas los circuitos lógicos presentan el mismo tamaño de transistor, $0{,}24/0{,}18$. Estos son los transistores que no aparecen numerados. $C_{n,1} = 2{,}4/2{,}4$ (Dimensiones en μm).

En primer lugar, hallamos el terminal al que se conectaría el electrodo P del fotodiodo, estando el N conectado a tierra, ya que cómo se comentó en el Capítulo 2 configuraremos este fotodiodo para que funcione en su región de polarización directa como celda solar. Así mismo, a este terminal encontramos conectado un transistor de *RESET*. Este descargará el fotodiodo hasta llevar su voltaje a la tensión de referencia cuando sea necesario (lo cual dependerá del modo de funcionamiento). Para la generación de esta señal de *RESET* del fotodiodo es empleado un circuito lógico que se explicará posteriormente.

Este mismo terminal está conectado a la entrada positiva del comparador, que será el encargado de emitir un pulso cuando el voltaje del fotodiodo supere un voltaje umbral, V_{th}. Para que no consuma de forma estática y emita pulsos únicamente cuando sea necesario, este comparador cuenta con transistores de *ENABLE* y de *RESET*. La señal de control de estos transistores (*RESET_COMP*, cómo se verá más adelante) variará según el modo en el que nos encontremos, y para su generación se implementa un Elemento C y un circuito lógico que también veremos más adelante.

Por otro lado se encuentran los transistores de pull-down conectados a las líneas de peticiones de filas y columnas, que serán los encargados de emitir estas solicitudes. Por compromiso entre el área en transistores y la funcionalidad del píxel, se han colocado líneas independientes para las peticiones del modo Octopus y del modo Continuo.

Además de esto, entre la salida del comparador y las líneas de peticiones del modo Octopus, encontraríamos implementada una celda de memoria cuya función es evitar pérdida de peticiones por retrasos en el protocolo.

3.2 Entradas y salidas

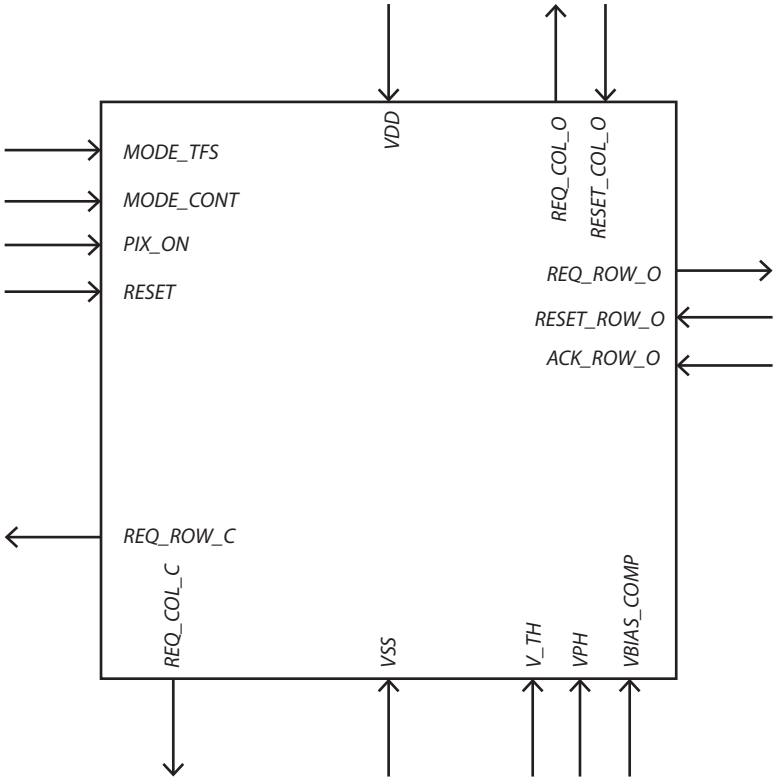

Figura 3.3 Entradas y salidas del píxel.

Para la comprensión del funcionamiento del píxel es esencial conocer sus señales de entrada y salida. En la Figura 3.3 se encuentran estas señales representadas, siendo las entradas:

- VDD y VSS. Tensión de alimentación y tensión de referencia, respectivamente.
- $MODE\_TFS$ y $MODE\_CONT$. Hacen referencia a las señales lógicas de control del modo de funcionamiento del píxel. Así, tendremos:

Tabla 3.1 Señales de control para modos de funcionamiento.

MODO	$MODE\_TFS$	$MODE\_CONT$
Free Running	0	0
Time to First Spike	1	0
Continuo	0	1

- PIX_ON y $RESET$. Son las señales de medida y reset que enviaremos desde el exterior.

- V_TH. Tensión umbral del comparador que determina a partir de que tensión en el fotodiodo consideramos un píxel iluminado.
- VPH. Nodo al que conectaremos el fotodiodo configurado como celda fotovoltaica.
- $VBIAS\_COMP$. Tensión de polarización del comparador.
- ACK_ROW. Señal emitida desde la periferia de reconocimiento de la petición por filas, que permitirá realizar la petición por columnas. (Sólo modo Octopus)
- $RESET\_ROW\_O$ y $RESET\_COL\_O$. Señales generadas por la periferia una vez se ha procesado la petición de fila/columna y que permiten bloquear el píxel (TFS) o resetear la celda de memoria (FR) cuando ambas están activas. (Sólo modo Octopus)

y las salidas:

- REQ_ROW_C, REQ_COL_C,REQ_ROW_O y REQ_COL_O. Son los nodos que conectaremos a las líneas de petición por filas y columnas del modo Continuo y el Octopus, respectivamente. Estos terminales estarán conectados además de a la periferia, a unos transistores de pull-up. De esta forma las señales serán activas nivel bajo: cuando el píxel deba emitir una petición, los transistores de pull-down internos serán los encargados de hacerlo, cambiando estas señales a un valor lógico bajo. Cuando la petición acabe, los pull-up externos devolverán las señales a su estado inactivo.

3.3 Principio de funcionamiento

Para describir el funcionamiento del píxel dividiremos la operación de éste en los diferentes eventos que irían ocurriendo secuencialmente en el tiempo. Hay que tener en cuenta que estos pasos variarán ligeramente dependiendo del modo de funcionamiento en el que nos hallemos. Se incluye un diagrama de flujo en la Figura 3.4 con el objetivo de hacer un primer acercamiento al funcionamiento del sistema. Se explicará detalladamente cada modo, la evolución de sus señales temporalmente y su funcionamiento concreto más adelante en cada sección de este capítulo.

1. Reset del fotodiodo. La tensión del terminal P del fotodiodo es llevada a la tensión de referencia por un transistor de RESET.
2. Integración de la carga. Los fotodiodos que estén captando energía por estar iluminados elevarán la tensión de su polo P con una velocidad proporcional al nivel de iluminación de cada píxel.
3. Disparo del comparador. Cuando la tensión del fotodiodo supera la tensión umbral la salida del comparador se activa.
4. Petición de lectura. Al activarse la salida del comparador se envía una petición de lectura por filas y columnas a la periferia.
5. Finalización del proceso. Generación de señal de reseteo del fotodiodo en el caso del FR y bloqueo del píxel en el TFS. En el modo continuo, ninguna acción sera necesaria.

Independientemente del modo en el que nos hallemos, para que se pase de la integración de la carga al disparo del comparador ha de estar activa la señal de medida PIX_ON.

En el modo Octopus, para que la integración de la carga se realice en el momento deseado, y no obtengamos una medida falsa por una captación de energía previa en el tiempo al momento en el que deseamos medir, la señal de RESET y la señal de medida que daremos al píxel estarán coordinadas. Es decir, hemos de asegurarnos que se emita desde la periferia una señal de reset unos instantes previos a la de medida, de forma que en el momento de comenzar a medir sea cuando se comience a cargar el fotodiodo y así podamos extraer del tiempo del primer pulso la intensidad luminosa. Si este reset fuese anterior al momento de pixon pero con un tiempo intermedio, los fotodiodos estarían precargados para el momento de pixon y el tiempo del pulso desde el momento del inicio de medida no sería proporcional a su intensidad luminosa. Habría que tomar en este caso como referencia el tiempo de reset pero para mayor simplicidad en los cálculos se sincronizan las señales a un mismo instante.

En el modo Continuo, como buscamos una señal de salida útil para el momento en el que se realice la medida, se le dará un tiempo de precarga al sistema para que al menos los fotodiodos de los píxeles más iluminados se encuentren activos. Así, si hay píxeles menos iluminados que necesiten un tiempo de carga

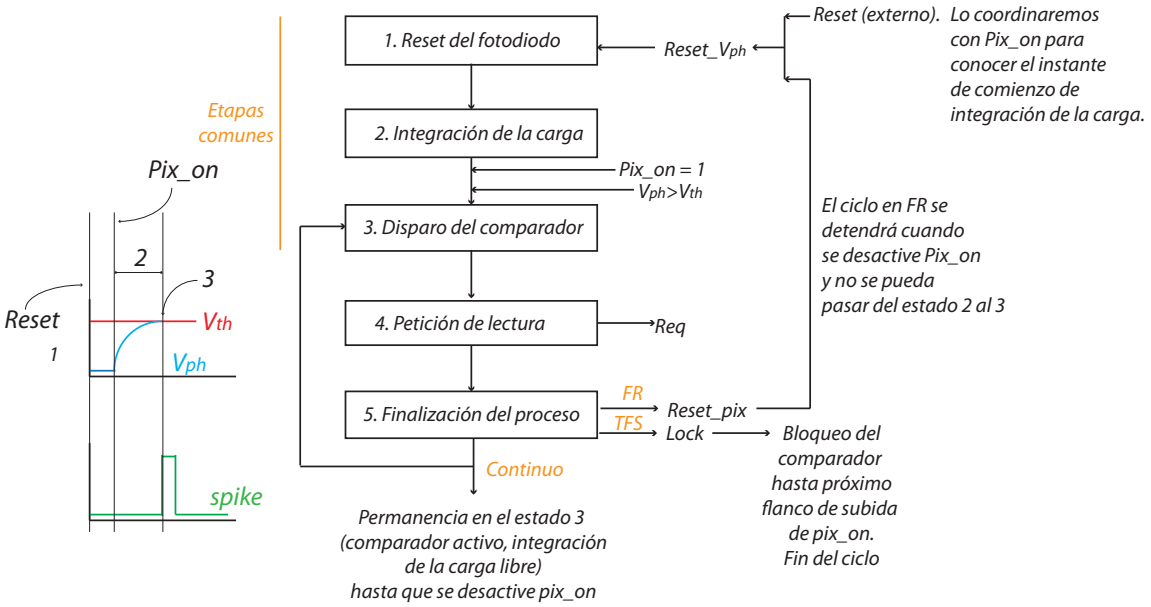

Figura 3.4 Diagrama de flujo ilustrativo de las etapas en el proceso de operación del píxel.

mayor, estos se activarán posteriormente, pero como ya se comentó la información más relevante sobre la posición del centroide la contienen los píxeles que generen mayores fotocorrientes por lo que esto no sería un problema. Para conseguir este tiempo previo de integración, el *RESET* ha de ser activado unos μs antes de comenzar la medida.

Cómo se definen e implementan cada una de estas señales será igualmente detallado junto con los resultados de la simulación de cada uno de estos modos en distintos escenarios, que se presentan en el Capítulo 5.

En esta sección se pretende describir el proceso de operación del píxel, y no entrar en profundidad en la implementación de cada elemento. Aquellos elementos que impliquen una suboperación y un diseño mas complejos serán explicados en la sección 3.4. Se incluyen a continuación en la Figura 3.5 cronogramas de las principales señales de cada modo para una mejor comprensión del funcionamiento de cada modo de operación.

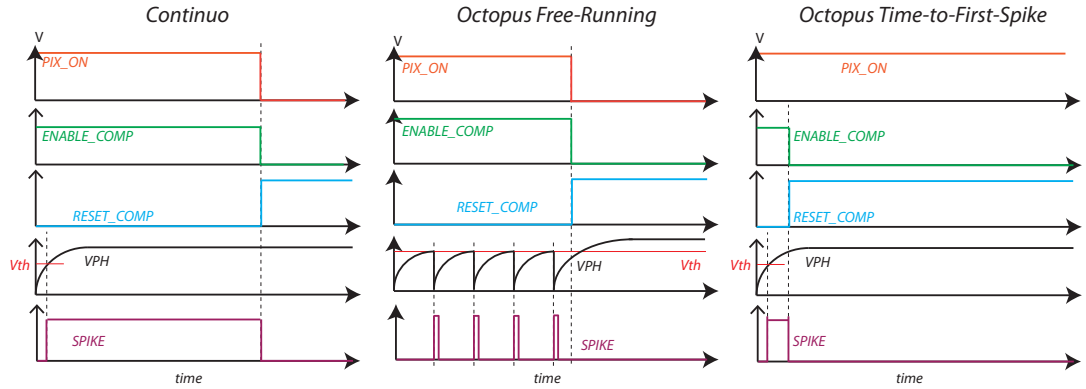

Figura 3.5 Cronogramas ilustrativos de las principales señales del píxel en cada modo de operación.

3.3.1 Reset del fotodiodo

Para que se realice la medida en el momento en el que deseamos tomarla, hemos de asegurarnos de que todos los fotodiodos son reseteados, es decir, la tensión de su cátodo y su ánodo son ambas llevadas a la

referencia, de forma que el fotodiodo no almacene carga. Para llevar a cabo esta operación se coloca en paralelo al fotodiodo un transistor de *RESET*, implementado como un transistor de tipo N cuya puerta está conectada a una señal *RESET_VPH* de forma que cuando esta señal esté activa nivel alto, el transistor conducirá la corriente y se llevará a cabo el RESET. La señal *RESET_VPH* se generará con un circuito lógico combinacional que coordina el RESET externo con, si fuera necesario, un RESET marcado por el modo de funcionamiento. Este circuito se verá en detalle a continuación, en el apartado 3.3.5.

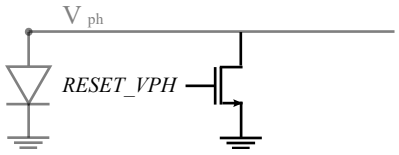

Figura 3.6 Transistor de RESET del voltaje del fotodiodo.

3.3.2 Integración de la carga

Una vez se ha precargado el fotodiodo a la tensión de referencia, se desactiva la señal de *RESET_VPH* de forma que se libera la tensión en el ánodo de éste. En este momento, los fotodiodos que se encuentren iluminados comenzarán a generar una fotocorriente, que irá cargando su capacidad interna. Esta carga, como ya se planteó en múltiples ocasiones anteriormente, y respondiendo a la naturaleza del comportamiento de los fotodiodos, será más rápida cuanto mayor sea el nivel de iluminación.

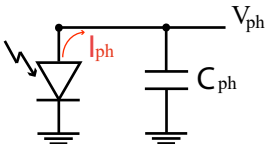

Figura 3.7 Esquema ilustrativo del fotodiodo iluminado durante el periodo de integración de la carga.

3.3.3 Disparo del comparador

Cuando al integrar la carga el fotodiodo supera una tensión umbral, especificada por la entrada V_{th}, la salida del comparador se activa. Será esta salida la que active, ya sea directa o indirectamente dependiendo del modo, el proceso de petición de lectura.

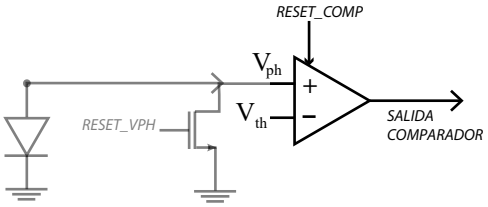

Figura 3.8 Símbolo, entradas y salidas del comparador.

3.3.4 Petición de lectura

En este paso del proceso encontramos necesidades diferentes dependiendo del modo de funcionamiento, motivo por el cual el diseño difiere y se implementaron líneas independientes para las peticiones del modo Octopus y el Continuo.

Petición de lectura en el modo Octopus

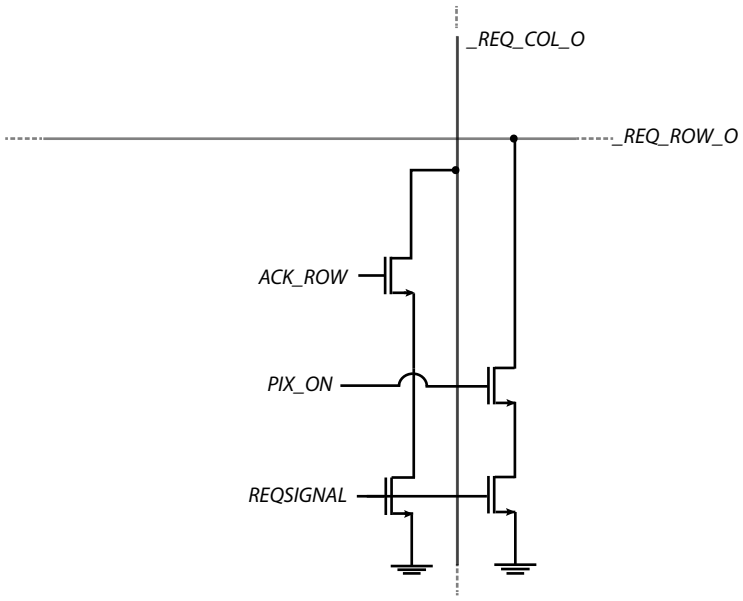

Figura 3.9 Líneas de petición del modo Octopus.

El modo Octopus se comunica con la periferia según el protocolo AER descrito en la sección 2.2.1 de este documento. El procesamiento de cada evento en este protocolo tomará un tiempo que puede causar que, si dos peticiones colisionan, una se pierda. Con el fin de evitar esto, se implementa una celda de memoria basada en un condensador que hará que se mantenga la petición un tiempo determinado si es que esta no está siendo recibida. Será la salida de esta celda de memoria, *REQSIGNAL*, la que conectemos a los transistores de pull-down encargados de enviar las peticiones a filas y a columnas.
La línea de peticiones por filas cuenta además con otro transistor de pull-down encargado de que sólo se puedan enviar peticiones cuando *PIX_ON* está activa, dado que se comprobó que en ausencia de esto podían darse glitches debido a la carga remanente en *REQSIGNAL*.
Por último, la línea de petición de columnas cuenta con un tercer transistor de pull-down activo únicamente cuando *ACK_ROW* sea activada por la periferia, lo cual implica que sólo se puede realizar la petición por columnas una vez se ha aceptado la de la fila. No será necesario incluir el transistor de *PIX_ON* en las peticiones por columnas gracias a éste transistor, dado que si no se ha emitido solicitud por filas, nunca se emitirá *ACK_ROW*, por lo que nos basta con incluirlo en una. Se incluye un esquema ilustrando qué señales comparten los píxeles según su fila y su columna en la Figura 3.10.

Petición de lectura en el modo Continuo

En este modo, dado que las peticiones por filas y columnas se realizan de forma continua (de ahí su nombre) siempre que la señal de medida esté activa, se estará realizando una petición en todo momento en que la salida del comparador esté activa. Esto se implementa conectando a las puertas de los transistores de pull-down de las líneas de peticiones la señal $SPIKE$.

3.3.5 Finalización del proceso

Una vez que todo el proceso de comunicación con la periferia ha sido llevado a cabo, es decir, que la transmisión de las coordenadas del píxel iluminado ha sido completada, el modo FR necesita que el voltaje a la entrada del comparador sea reseteado para seguir enviando peticiones, y el TFS que el píxel se "bloquee" para que no siga ni emitiendo peticiones ni consumiendo corriente. En el modo continuo, no buscamos modificar el voltaje del fotodiodo, sino que éste evolucione durante la medida según la iluminación. Recordamos que en la figura Figura 3.12 se representa cómo se desea que evolucionen V_{PH} y $SPIKE$ durante un periodo de medida

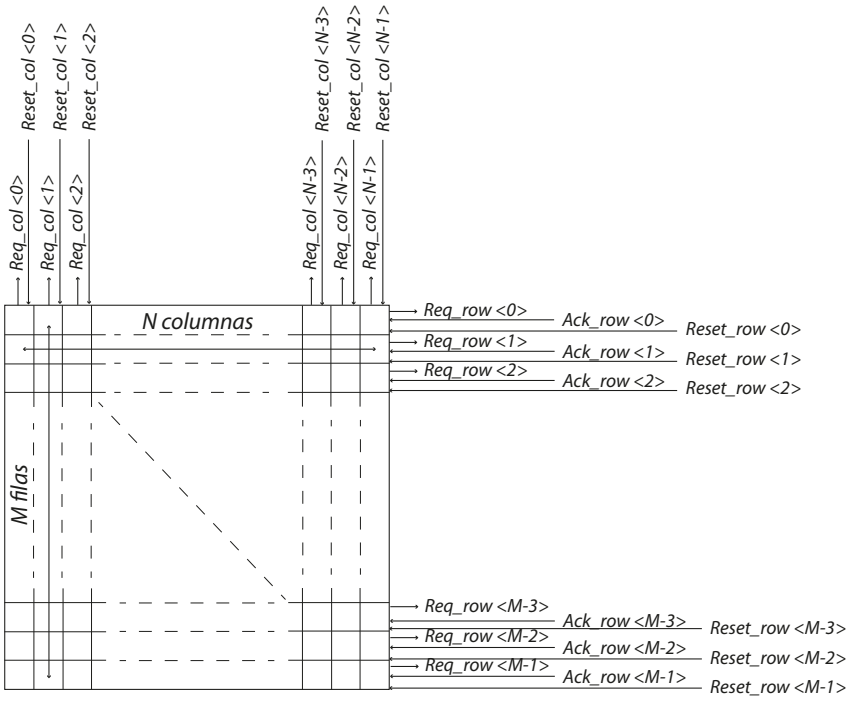

Figura 3.10 Esquema ilustrativo de la distribución de señales compartidas por filas y columnas en el sensor Octopus.

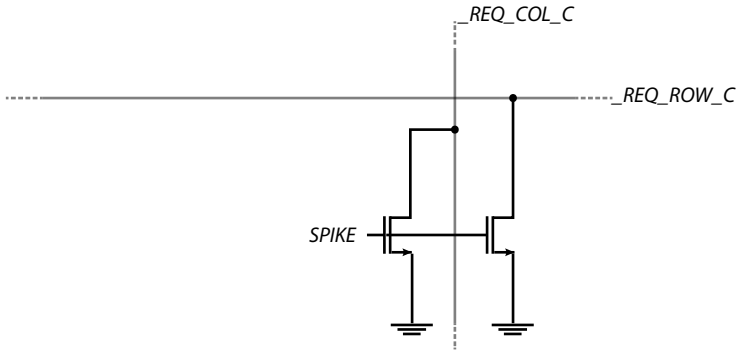

Figura 3.11 Líneas de petición del modo Continuo.

en función del tiempo.

Así, esto será logrado en cada modo según se describe a continuación:

Modo FR. Generación de *RESET_VPH*

Como se comentó con anterioridad, el terminal conectado al comparador de la tensión del fotodiodo tiene un sólo transistor de RESET. Esto se hace entre otras cosas porque si se colocasen varios transistores en paralelo (con la intención de poder resetear con señales independientes) las corrientes de fuga que estos transistores tendrían debido a sus cargas parásitas podrían llegar a anular el funcionamiento correcto del comparador, debido a que las corrientes que genera el fotodiodo como celda solar son muy pequeñas. Como consecuencia surge la necesidad de generar una señal única, que es la llamada $RESET\_VPH$, que controle el reseteo del fotodiodo, coordinando el RESET externo con el interno, marcado por cada modo.
Así, esta señal será la salida de la función lógica:

$$RESET\_VPH = RESET + (SPIKE \cdot \overline{MODE\_TFS} \cdot \overline{MODE\_CONT})$$

Es decir, el fotodiodo será reseteado, o bien cuando se active la señal externa de RESET (necesario en los tres modos), o bien cuando, estando en modo FR, se produzca un SPIKE (lo que indicaría que se ha emitido una petición y por tanto hay que volver a medir). De esta forma, este RESET reiniciaría el proceso descrito de operación para el modo FR. El circuito electrónico que resulta de la implementación de esta lógica sería el siguiente:
Y responde a la tabla de verdad que encontramos a continuación.

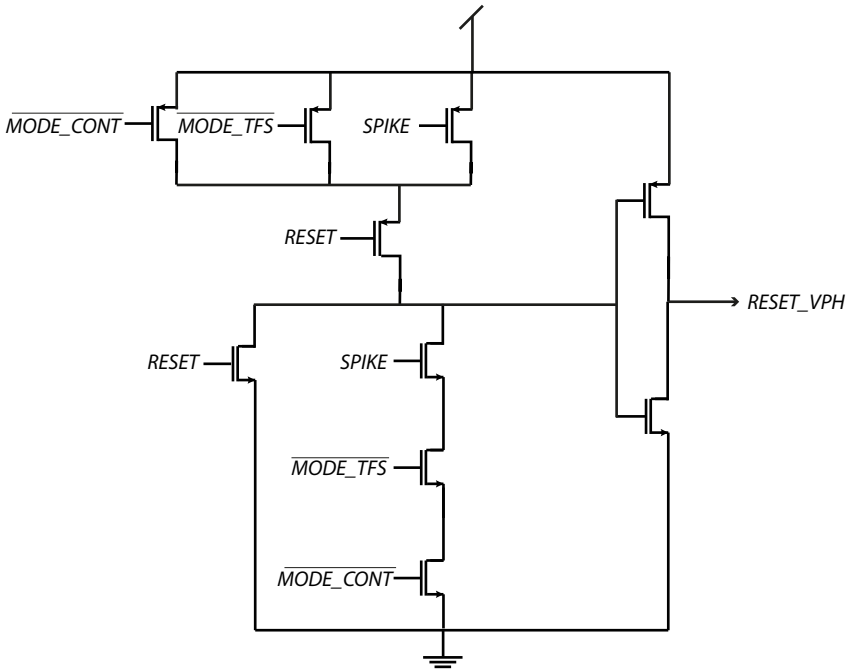

Figura 3.12 Esquemático del circuito para la generación de $RESET\_VPH$.

Tabla 3.2 Tabla de verdad del circuito lógico de generación de $RESET\_VPH$.

Estado del proceso	RESET	$\overline{MODE\_TFS}$	$\overline{MODE\_CONT}$	$SPIKE$	$RESET\_VPH$
Modo TFS	0	0	-	-	0
Modo Continuo	0	-	0	-	0
Modo FR: inactividad/ entre medidas	0	1	1	0	0
Modo FR: pulso en SPIKE	0	1	1	1	1
RESET externo	1	-	-	-	1

Modo TFS. Bloqueo del píxel

Cuando el píxel trabaja en modo TFS lo que buscamos es que se emita un solo pulso, y por tanto, una vez se ha activado la salida del comparador, no deseamos que esto vuelva a ocurrir. Así mismo, para ahorrar consumo de potencia, querremos que la salida del comparador quede a un nivel lógico bajo, con un consumo estático mínimo. Esto se consigue mediante el control de las señales ENABLE y RESET del comparador, y se verá con detalle en la sección dedicada al estudio de éste. El resultado que se alcanza es que cuando trabajamos en TFS, el comparador quede reseteado una vez se ha emitido la primera petición del píxel. No volverá a activarse hasta que no se desactive y se vuelva a activar la señal de medida.

Modo continuo

Es necesario comprender que el principio del modo continuo es que en todo momento durante el periodo de medida los píxeles iluminados estén emitiendo peticiones. Por eso mismo, en este modo no se generará un

funcionamiento cíclico como en el FR ni se bloqueará el píxel como en el TFS, sino que se dejará la salida del comparador "libre" a ir reaccionando a los cambios en el voltaje del fotodiodo producidos por la variación de la iluminación del sensor. Esto implica que cuando un píxel reciba radiación, tardará un tiempo en activarse, t_{carga}, puesto que la carga no es instantánea, y cuando deje de ser iluminado, transcurrirá $t_{descarga}$, hasta que el comparador desactive su salida, puesto que la capacidad del fotodiodo tarda un tiempo en descargarse.

3.4 Profundización en la estructura de ciertos elementos

En esta sección se explicará con más detalle el funcionamiento y diseño de ciertos elementos empleados en el diseño del píxel que presentan un funcionamiento que no ha quedado completamente detallado anteriormente.

3.4.1 Comparador

Un comparador no es más que un amplificador operacional configurado de tal forma que compare sus dos entradas y genere una salida en función de esta comparación y de la aplicación que se haya buscado al diseñarlo. En nuestro caso, se optó por un comparador con el siguiente esquemático:

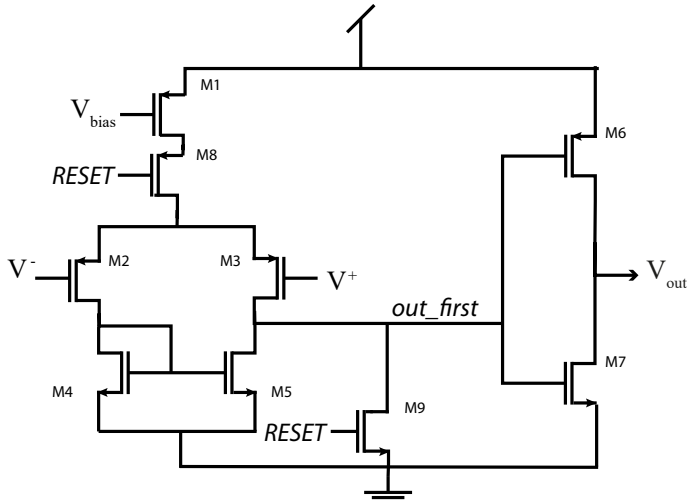

Figura 3.13 Esquemático del comparador. $M_{n,1} = 7{,}04/0{,}4$, $M_{n,2,3} = 0{,}24/0{,}18$, $M_{n,4,5} = 0{,}88/2$, $M_{n,6,7} = 0{,}44/0{,}3$ $M_{n,8} = 7{,}04/0{,}4$, $M_{n,9} = 0{,}88/0{,}3$. (Dimensiones en μm).

El objetivo de éste es comparar la tensión en el nodo de entrada positivo (V^+) con la tensión en el nodo negativo (V^-), y si la primera es mayor emitir una señal de salida de nivel lógico alto. (Señal que permanecería en nivel lógico bajo mientras que V^+ no superase V^-).

Para ello, se implementó un par diferencial compuesto por dos transistores MOSFET tipo P (M2,M3) cuyas puertas estarían conectadas a las dos entradas, positiva y negativa. Éste par se encuentra polarizado por otro transistor tipo P (M1) que generalmente se halla funcionando como fuente de intensidad polarizada mediante un espejo de corriente.

Así, conectamos a los drenadores de este par un espejo de corriente (compuesto por los transistores M4 y M5) que funciona como carga y consigue una salida single-ended que sea la comparación de ambos terminales. Es vital la colocación de este espejo, para conseguir que en la salida out_first la intensidad sea positiva solo cuando $V^+ > V^-$ y así, esta señal tome un valor lógico alto cuando el voltaje en el fotodiodo supere el valor de la tensión umbral V_{th}.

Esta salida se conecta a una etapa lógica inversora, formada por los transistores M6 y M7, para amplificarla, puesto que se busca que la señal lógica de salida (V_{out}) pertenezca al rango $[0{,}1.8]V$. En consecuencia, la señal de salida será en realidad $V_{out} = \overline{SPIKE}$.

Por último, encontramos un transistor de *ENABLE* (M8) y uno de *RESET* (M9), que nos permitirán controlar con señales externas si el comparador está activo o no. Cómo podemos observar, en realidad no encotramos una señal de *ENABLE* de entrada al comparador cómo tal, a pesar de que hemos estado mencionando esta funcionalidad en todo el documento previamente. La realidad es que las señales de *ENABLE* y *RESET* son la misma pero negadas, y cómo en este caso el transistor de *ENABLE* tenía que ser de tipo P, la señal que se debe conectar a su puerta es la que se desea que lo active pero negada, de forma que se conecta directamente *RESET_COMP*. A pesar de ello, tanto anteriormente en este Trabajo Fin de Grado como a continuación, se seguirá haciendo referencia a la señal *ENABLE_COMP*, puesto que facilita la comprensión directa de cuándo se encuentra activo el comparador y cuándo no.

Generación de *RESET*

Se introdujeron en la sección 3.3 las diversas necesidades de diseño que introducía el hecho de poder conmutar el modo de funcionamiento del píxel. En lo relativo al comparador, nos encontramos con que hay que lograr:

1. Que el comparador sólo active su señal de salida cuando se encuentre activa la señal de medida *PIX_ON*, puesto que si no tendríamos un consumo estático innecesario y además se estarían emitiendo peticiones no deseadas.
2. Que en el modo Continuo y el FR, el comparador esté activo durante todo el periodo de medida, mientras
3. Que en el modo TFS, buscamos que el comparador emita un sólo pulso de salida y quede bloqueado, de forma que deje de consumir potencia y además el píxel no pueda realizar más peticiones durante ese ciclo de medida.

Para conseguir satisfacer estas especificaciones, se configuró la señal *RESET_COMP*, que será la señal conectada a las puertas de los transistores de *ENABLE* y *RESET* mencionados anteriormente.

Con el fin de comprender esta etapa del circuito, es necesario hacer un breve paréntesis y recordar que en el modo Octopus la periferia activará *RESET_ROW*/*RESET_COL* cuando haya terminado de procesar la petición de esa fila/columna, y por tanto cuando ambas estén activas en un píxel se podemos considerar que el proceso de petición de éste está completo y que, en TFS, el píxel ya puede ser bloqueado. Es para esto que mediante una puerta NOR cuyas entradas sean estas señales negadas se genera la señal *RESET_PIX*, indicadora de que la petición ha sido procesada. Se emplea esta lógica en concreto (representada en la Figura 3.14) porque será la que disminuya el área empleada en el píxel para la generación de la señal.

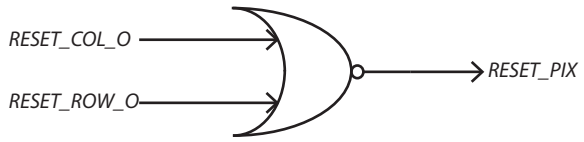

Figura 3.14 Esquemático del circuito de generación de *RESET_PIX*.

Continuando con la señal de *RESET*, el esquemático del circuito empleado para su generación sería el siguiente:

En primer lugar, encontramos un Elemento C, mediante el cual conseguiremos satisfacer la especificación 3, relativa al funcionamiento en TFS. La función de un Elemento C en electrónica es generar una salida que sólo cambie su estado cuando sus dos entradas toman el mismo valor. Así, cuando ambas estén activas, la salida será activa; cuando ambas estén desactivadas la salida estará desactivada, y el resto de estados serán estados de memoria.
Este elemento se halla seguido de un inversor, puesto que nos interesa que la señal generada (llamada *LOCK*, por su función de controlar el bloqueo del píxel) sea activa nivel alto, y la salida natural del Elemento C es una señal activa nivel bajo.
La tabla de verdad asociada al conjunto de elementos resultante, que encontramos relacionada con cada etapa del proceso de medida, es la mostrada en la Tabla 3.3.

Una vez se ha generado la señal *LOCK*, que determina cuando ha de bloquearse el píxel en TFS, se implementa una función lógica que consiga cumplir el resto de especificaciones definidas. Esta función sería

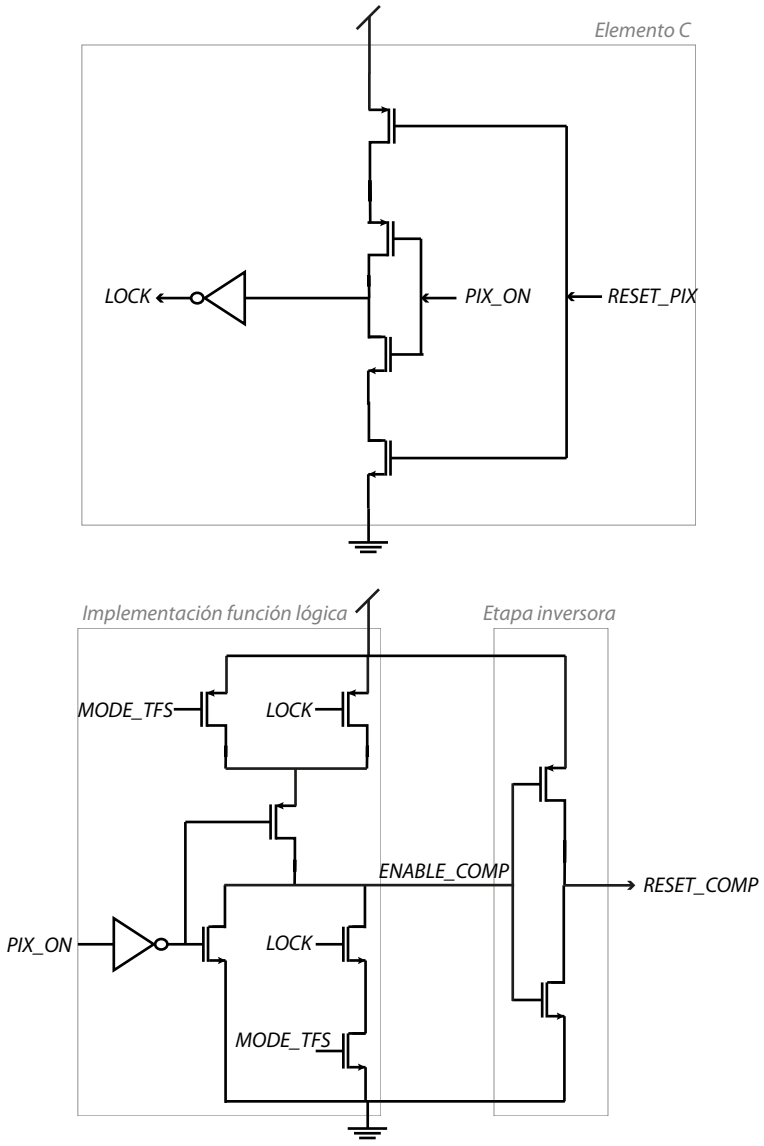

Figura 3.15 Esquemático del circuito de generación de $RESET\_COMP$.

Tabla 3.3 Tabla de verdad del Elemento C. Tras el bloqueo, vuelve al estado 1 de inactividad cuando se desactive la señal *Pix_on*.

Estado del proceso	PIX_ON	$RESET\_PIX$	$\overline{LOCK}$	$LOCK$
1. Inactividad (y_0)	0	0	0	1
2. Inicio medida	0	1	0	1
3. Petición procesada	1	1	1	0
4. Bloqueo	1	0	1	0

la siguiente:

$$ENABLE\_COMP = PIX\_ON \cdot (\overline{LOCK} + \overline{MODE\_TFS})$$

Con esta función lo que conseguimos es activar el comparador siempre que la señal de medida esté activa y no se esté en modo TFS, o estando en modo TFS, siempre que no haya necesidad de bloquear el píxel.

Complementariamente, la señal *RESET_COMP* será la negada de *ENABLE_COMP*, osea que desactivaremos el comparador o bien cuando no se esté midiendo o bien cuando el píxel haya de ser bloqueado en modo TFS. Conseguimos esta señal por tanto colocando una etapa inversora a la salida del circuito lógico que genera el *ENABLE*. Cómo ya se comentó anteriormente, en realidad la señal *ENABLE_COMP* cómo tal no será empleada, ya que para la implementación física sólo necesitamos su negada. Sin embargo, comprender su función lógica y explicación resulta esencial para comprender dicha implementación y la señal *RESET_COMP*.

La tabla de verdad a la que por tanto responderá este circuito lógico es la que encontramos en la figura Tabla 3.4 . Se han hecho anotaciones sobre el momento del proceso en el que se da ese estado para mejor comprensión.

Tabla 3.4 Tabla de verdad del circuito lógico de generación de RESET y ENABLE del comparador.

Estado del proceso	*MODE_TFS*	*PIX_ON*	*LOCK*	*ENABLE_COMP*	*RESET_COMP*
Inactividad	0	0	0	0	1
- (No ocurre)	0	0	1	0	1
Inicio medida	0	1	0	1	0
Medida completa	0	1	1	1	0
Inactividad	1	0	0	0	1
- (No ocurre)	1	0	1	0	1
Inicio medida	1	1	0	1	0
Medida completa	1	1	1	0	1

Al respecto de esta tabla habría que mencionar que en el modo Continuo la señal *LOCK* se mantiene siempre a 0 al no generarse *RESET_PIX* (esta se mantiene inactiva). Por tanto, en este modo pasaríamos solo por los estados de "Inactividad" e "Inicio medida". Como podemos observar también, en el modo FR se pasa por el estado de "Medida Completa" pero en este caso no afectará al comparador, dado que queremos que este siga funcionando. Este cambio de estado solo afectará al *RESET_VPH*, tal y como se explicó anteriormente en este Trabajo Fin de Grado en la sección 3.3.5.

3.4.2 Celda de memoria

Anteriormente, en la sección 3.3.4, se mencionó la necesidad de incorporación de un elemento de memoria a la salida del comparador para el correcto funcionamiento en modo Octopus de la emisión de peticiones.

Éste elemento será una celda de memoria compuesta por una capacidad de tipo MIM-Cap (*Metal-Insulator-Metal Capacitor*), capacidad generada empleando las dos últimas capas de metal del circuito integrado. Resulta interesante el empleo de este tipo de capacidades dado que pueden integrarse sin consumir área activa en el chip. El tamaño de esta capacidad ($C_{n,1} = 2{,}4/2{,}4$) y el de sus transistores de *RESET* ($M_{n,2,3} = 4/0{,}18$) fue diseñado para conseguir un tiempo de permanencia de la petición del orden del milisegundo. Teniendo en cuenta que la periferia tiene un tiempo de procesamiento del orden de nanosegundos, si en 1 milisegundo la petición no ha sido procesada es porque ha habido un error no relacionado con el colapso de peticiones y por tanto no nos interesa mantener la petición más tiempo.

Encontramos el esquemático de la parte del circuito correspondiente a la celda de memoria en la figura Figura 3.16. En esta podemos ver que en paralelo a la capacidad, se añaden dos transistores de RESET, dado que esta celda ha de reiniciar su valor a la tensión de referencia o bien cada vez que se resetee la matriz externamente antes de realizar una medida (transistor M1 cuya puerta está conectada a la señal *RESET*) o bien cuando ya se haya recibido la última petición emitida, evento señalizado por la activación de *RESET_PIX*, que implicaría que este valor ha de ser actualizado (transistor M3 cuya puerta está conectada a *RESET_PIX*).

Así, cada vez que se produzca un pulso a la salida del comparador, su señal negada activará la circulación por el transistor M1 de la figura Figura 3.16. Esto hará que se cargue la capacidad. De esta forma la señal que enviaremos a los transistores de pull-down de las líneas de petición será el voltaje en el nodo capacitivo, señal a la que denominamos *REQ_SIGNAL*. Vemos un cronograma representativo del funcionamiento de la celda de memoria en Figura 3.17.

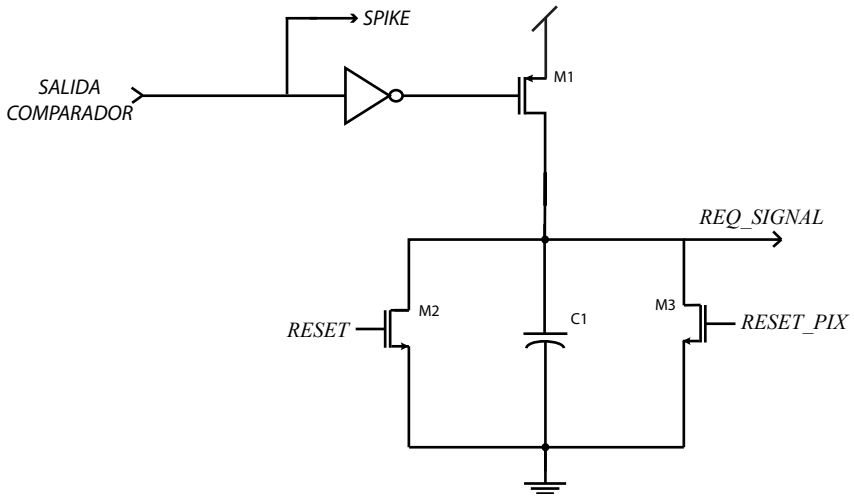

Figura 3.16 Esquemático de la celda de memoria.

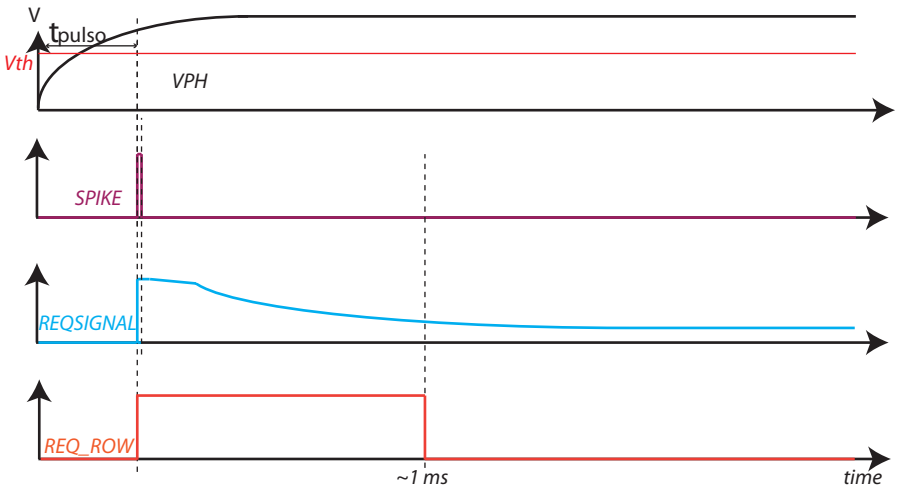

Figura 3.17 Cronograma del funcionamiento de la celda de memoria.

4 Estructura de los circuitos de lectura periféricos

El propósito principal de los sensores solares para el control de la actitud en navegación espacial es la determinación del centroide del área iluminada del sensor (*ROI*), información a partir de la cual se conseguirá determinar la posición del Sol respecto al mismo.

Tal y cómo se planteó en el Capítulo 2 de este documento, dependiendo del modo de funcionamiento en el que nos hallemos, la información que se transmite del sensor al exterior es distinta, al ser distinto el modo en el que se calculará el centroide. Es por ello que el modo Octopus y el modo Continuo requieren periferias distintas, entendiéndose la periferia del sensor como toda la circuitería que forma parte de éste además de la matriz de píxeles. La periferia del modo Octopus emitirá como salida un vector de intensidades luminosas codificadas en la frecuencia, y la del modo Continuo aportará directamente las coordenadas del centroide. En este capítulo se pretende presentar la estructura de cada una de ellas.

4.1 Periferia del canal de lectura sensor en modo de operación Octopus

Si describimos la arquitectura completa de un sensor Octopus, los elementos que lo componen son:

- Matriz de píxeles de *M* filas y *N* columnas. La fila a la que pertenece un píxel sería su coordenada Y y su columna la coordenada X.
- Bloques periféricos encargados de procesar las señales de salida de la matriz y comunicarse con ella mediante las señales de entrada. A su vez, se comunican con el exterior (en este caso consideramos que la unidad de procesamiento es una FGPA, que será la encargada del cálculo del centroide) mediante un bus de datos, una línea de petición de envío de datos y una línea de señal de recepción de datos.

Podemos encontrar una representación esquemática de esta arquitectura en la Figura 4.1, dónde hallamos la periferia dividida en sus distintos bloques. Hay que tener en cuenta que tanto en esta representación como en las siguientes de este capítulo, cada vez que se menciona una señal (ejemplo:*_REQ*) se esta haciendo referencia al vector que contiene las señales de cada una de las filas/columnas de la matriz. En esta figura podemos observar que se divide la periferia en bloques de forma que quedan dos conjuntos idénticos, uno encargado del procesamiento de las señales por filas y otro por columnas.

A continuación, explicaremos brevemente el funcionamiento de cada bloque, centrándonos en qué señales reciben cómo entradas y cuáles emiten como salidas. Dado que el objetivo de este capítulo no es la explicación detallada del funcionamiento a nivel de componentes y de diseño, sino la comprensión del funcionamiento de la periferia y la comunicación de ésta con la matriz de píxeles y el exterior, en cada bloque no se describirá cada elemento que lo conforma (inversores, transistores, puertas lógicas...) sino las sus funcionalidades de los conjuntos.

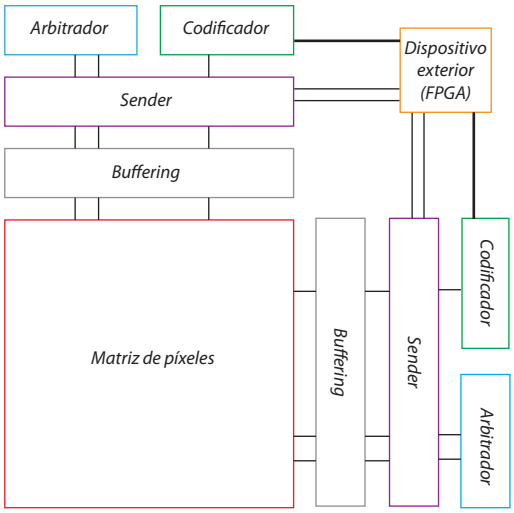

Figura 4.1 Esquema de la arquitectura del sensor Octopus, compuesto por la matriz de píxeles y los bloques que conforman la periferia.

4.1.1 Buffering

Nos encontramos trabajando con un sistema asíncrono, en el que la aparición de glitches puede causar fallos que ocasionen desde la corrupción de la información hasta la degeneración total del funcionamiento del sensor.
Por otra parte, el hecho de que las señales que se emplean para la entrada y salida del píxel sean compartidas por un gran número de estos (las señales son compartidas por toda una fila/columna) implica la existencia de capacidades parásitas elevadas debidas a la propia longitud de la línea y la conexión de múltiples transistores a éstas. Esto genera unos tiempos de subida y bajada en las señales relativamente elevados. Estas dos condiciones causan que sea altamente recomendable el uso de buffers, que serán etapas encargadas de regenerar las señales digitales, logrando evitar glitches debidos a los elevados tiempos de subida y bajada y obtener niveles lógicos bien definidos.

Esta etapa de buffering estará compuesta de dos inversores en serie por cada línea, cuyos tamaños se diseñarán en función de las capacidades características de cada línea. Podemos ver en la Figura 4.2 un esquema ilustrativo del bloque.

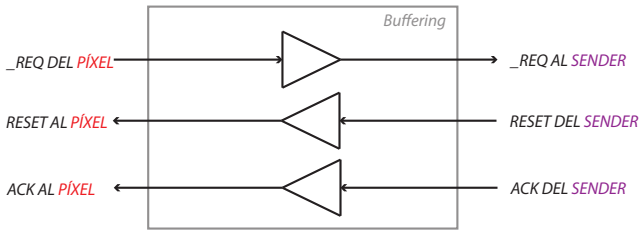

Figura 4.2 Esquema del bloque de buffering de la periferia del sensor Octopus.

4.1.2 Bloque para la comunicación según el protocolo AER

Este bloque recibirá también el nombre de *sender* por ser el encargado de comunicar la matriz y el resto de bloques de la periferia entre sí.

La estructura de este bloque es la reflejada en la Figura 4.3. El sender recibe las peticiones (*_REQ*) de la matriz de píxeles, y las envía al arbitrador para que éste otorgue la prioridad. Así, recibe del arbitrador el *ACK* y envía esta señal al codificador, a la vez que emite la petición para comunicar esa fila/columna con el

bus. (REQ_BUS).

Así mismo, es en éste bloque donde encontramos la circuitería encargada de generar $RESET$ (Que serán $RESET\_COL$ o $RESET\_ROW$ cuando lleguen al píxel). Cómo ocurre esto se encuentra más detallado en la sección correspondiente al codificador, en la que se explica el proceso de carga de información al bus de datos.

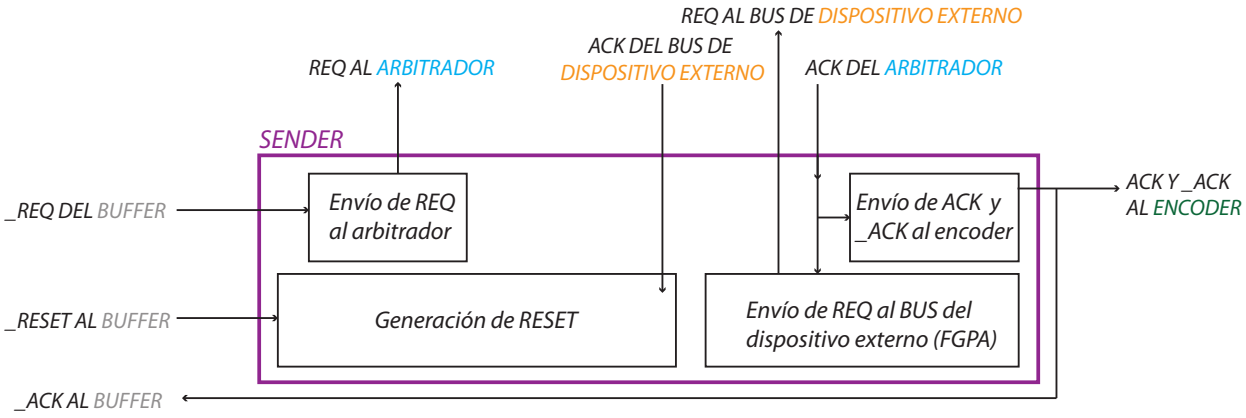

Figura 4.3 Esquema de las funciones y las señales de entrada y salida del bloque de comunicación AER de la periferia del sensor Octopus.

Llegados a este punto del documento, se han expuesto todos los elementos y procesos que tendrán lugar dentro del protocolo AER para este diseño de píxel. Para ilustrar cómo se desarrolla esta comunicación, se incluye el cronograma tentativo de la Figura 4.4, que representa cómo evolucionarían las señales una vez se activa PIX_ON en un píxel iluminado y en uno que no lo esté (tómese este último como referencia del estado inactivo de cada señal).Este diagrama representa el proceso para un pulso a la salida del comparador. Si el sensor se halla funcionando en modo FR, simplemente se repetirá todo el protocolo cada vez que se produzca un pulso en $SPIKE$, no generándose el LOCK. Para una mejor comprensión del protocolo, se desglosa a continuación las etapas que comprenderían el proceso:

1. Las peticiones son realizadas.
2. A nivel de fila, el circuito de arbitraje periférico (Arbitrador) da prioridad a una de ellas, activando la señal ACK_ROW (bandera de señal recibida) y $RESET\_ROW$ (indica que esa fila ya puede resetearse al haberse procesado su petición).
3. La activación de ACK_ROW permite que se realice el mismo proceso a nivel de columnas (sólo para las columnas activas en la fila priorizada).
4. A nivel de columna, el arbitrador prioriza una de las peticiones y se procesa la petición del píxel en el periférico a nivel de columna, activando el $RESET\_COL$ de esa columna. Al hallarse activas las dos señales de $RESET$ del píxel, se activa $RESET\_PIX$. En TFS el píxel se bloquea ($LOCK = 1$) y en FR se reinicia el protocolo AER para ese píxel.

4.1.3 Arbitrador

Este bloque será el encargado de otorgar la prioridad a las peticiones a la hora de comunicar los datos al bus del dispositivo externo, puesto que es común que cuando operamos de forma asíncrona aparezcan peticiones nuevas antes de haber procesado la última petición aceptada en el bus o incluso se den simultaneidades de peticiones en el tiempo. El arbitrador está compuesto de $n/2$ etapas (siendo n el número de entradas) formadas por arbitradores greedy (que siempre dan prioridad a la misma entrada en caso de colisión) estando comunicadas estas etapas entre sí en forma de árbol (distribución ilustrada en la Figura 4.5 (b)). La entrada del bloque serán las peticiones de la matriz de píxeles, ya sea por filas o columnas, y sus salida serán las líneas dedicadas a aceptarlas, estando en cada momento solo única de ellas activa, que será la asociada a la petición que haya llegado antes en el tiempo. Encontramos la representación del bloque en la Figura 4.5 (a).

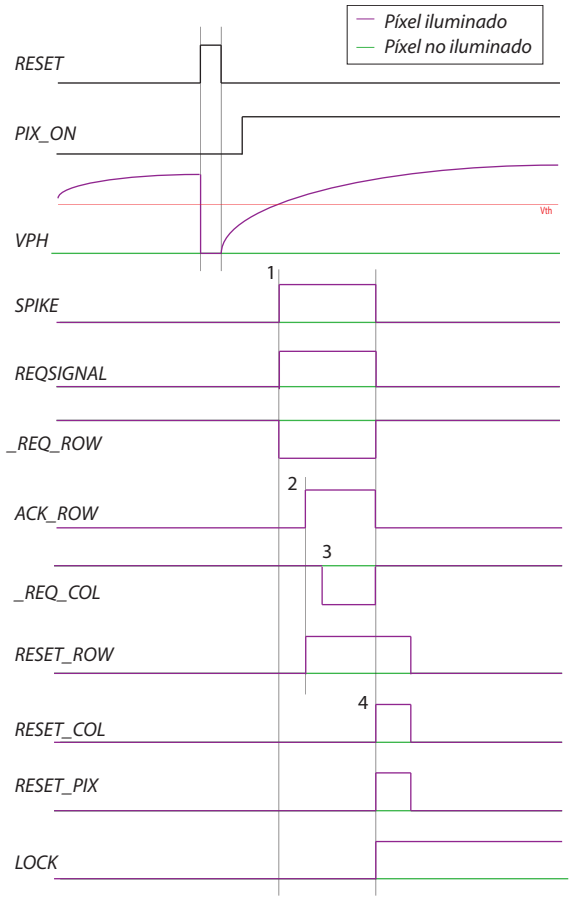

Figura 4.4 Cronograma de las señales involucradas en el protocolo AER y la operación del píxel del sensor en modo Octopus.

4.1.4 Codificador

El codificador (o *encoder*), cómo su propio nombre indica, es el encargado de generar las señales que contendrán la información de la dirección del píxel al que el arbitrador ha otorgado prioridad. Cuando un píxel de posición (i, j) ha realizado una petición que ha sido aceptada en ambas dimensiones, en cada una de ellas se activará la señal *ACK*. Como sería ineficiente emplear el vector de estas señales como código, puesto que tendría tantos elementos como el número de filas y columnas, es necesario emplear el *encoder* para codificar el código obtenido en un número menor de bits (Codificamos de $one-hot$ a binario). Así, es este el elemento que se halla conectado al bus de datos que comunica la periferia con el exterior. La estructura general de un *encoder* para un número N de entradas sería la que vemos en la Figura 4.6.

A medida que se acepta una petición por fila o columna, y esto llega en forma de *ACK* activo del *sender* (que los habrá recibido a su vez del arbitrador, cómo ya se vio anteriormente), el codificador varía su salida para transmitir por cada bit del bus de datos el valor que corresponda a la línea que está activa. De esta forma, a través del bus de datos la periferia emitirá como salida un vector de direcciones donde cada elemento se corresponde con un instante temporal. Es decir, podemos saber en qué momento se ha activado cada píxel, y asociar con la unidad de procesamiento ese tiempo que ha tardado en activarse a una intensidad luminosa, tal y cómo se explicaba en el Capítulo 2 de este Trabajo Fin de Grado.

Una vez se haya transmitido la información del píxel que se encuentra como prioritario en el arbitrador, el dispositivo externo emitirá una señal de *ACK* del bus, y en consecuencia se generará el *RESET* del sender, señal que llegará al píxel como *RESET_ROW* o *RESET_COL*, para que ese píxel se desactive y continúe el protocolo.

La estructura general de un *encoder* para un número N de entradas sería la que vemos en la Figura 4.6.

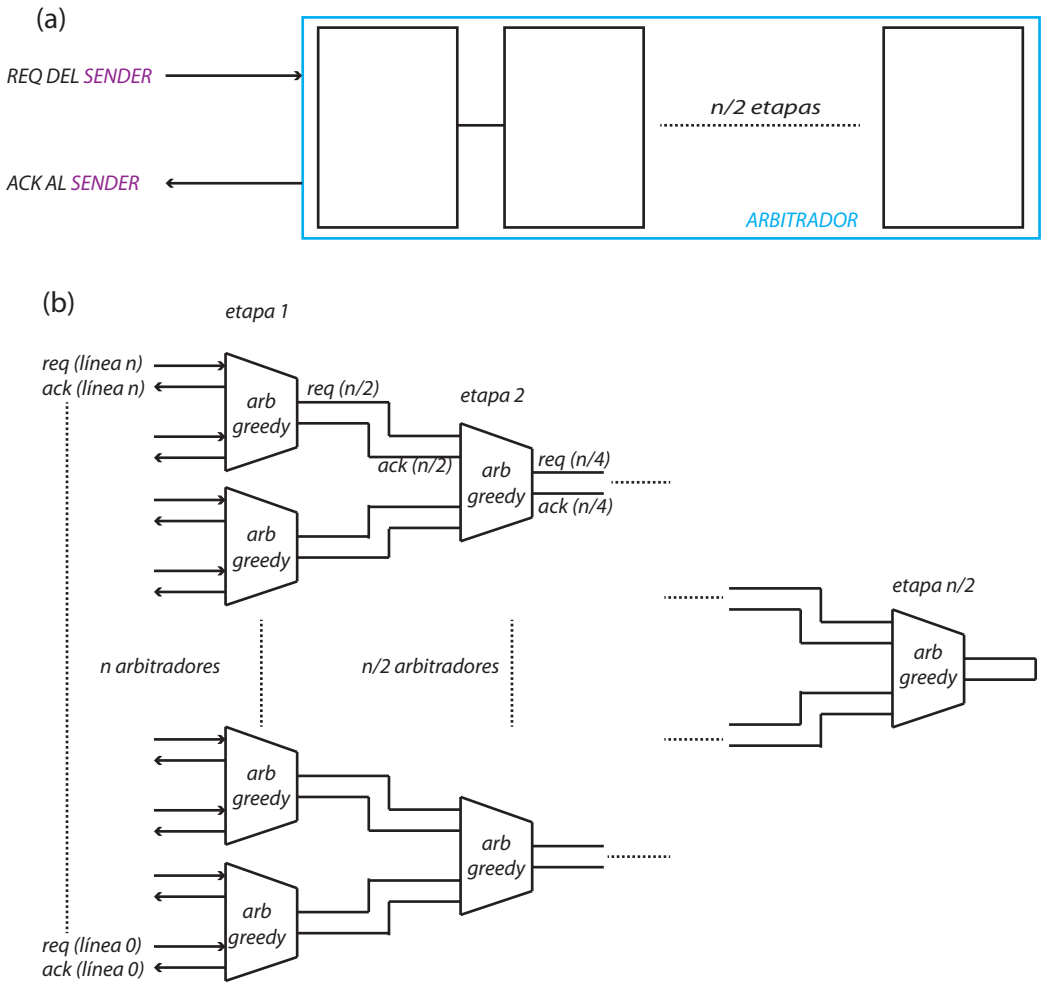

Figura 4.5 (a) Esquema de las funciones y las señales de entrada y salida del bloque de arbitradores de la periferia del sensor Octopus. (b) Esquema de la conexión de los arbitradores en árbol.

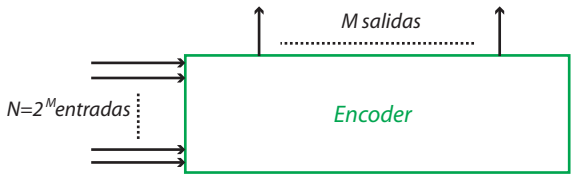

Figura 4.6 Esquema de las entradas y salidas de un *encoder* genérico.

De forma ilustrativa se incluye la figura Figura 4.7, ejemplo de un bloque de encoder con cuatro entradas y por tanto dos salidas. Este caso sería el de una matriz de 4×4 (Para codificar $4 = 2^2$ direcciones necesitamos 2 bits de información).

4.2 Periferia del canal de lectura del sensor en modo de operación Continuo

Un sensor Continuo estaría compuesto por los siguientes elementos:

- Matriz de píxeles de N filas y M columnas. La fila a la que pertenece un píxel sería su coordenada Y y su columna la coordenada X.

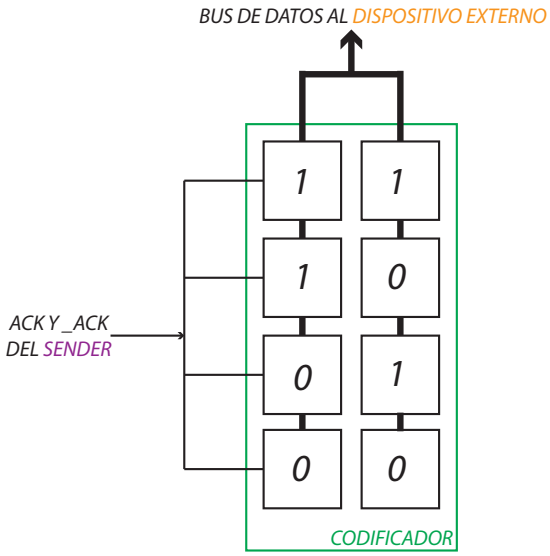

Figura 4.7 Ejemplo del esquema de un codificador para 4 entradas en la periferia del Sensor Octopus.

- Bloques periféricos encargados de procesar las señales de salida de la matriz y proporcionar en cada instante de tiempo la posición del centroide como señal de salida del sensor.

Encontramos una representación esquemática de esta arquitectura en la Figura 4.8. Podemos ver que al igual que en la arquitectura del sensor Octopus, hay dos conjuntos idénticos de periferia para las señales de salida columnas y filas de la matriz de píxeles. En este caso, la salida que obtendremos de la periferia serán directamente las coordenadas X e Y del centroide de la región iluminada, sin necesidad de un dispositivo externo que lleve a cabo el cálculo de esta posición.

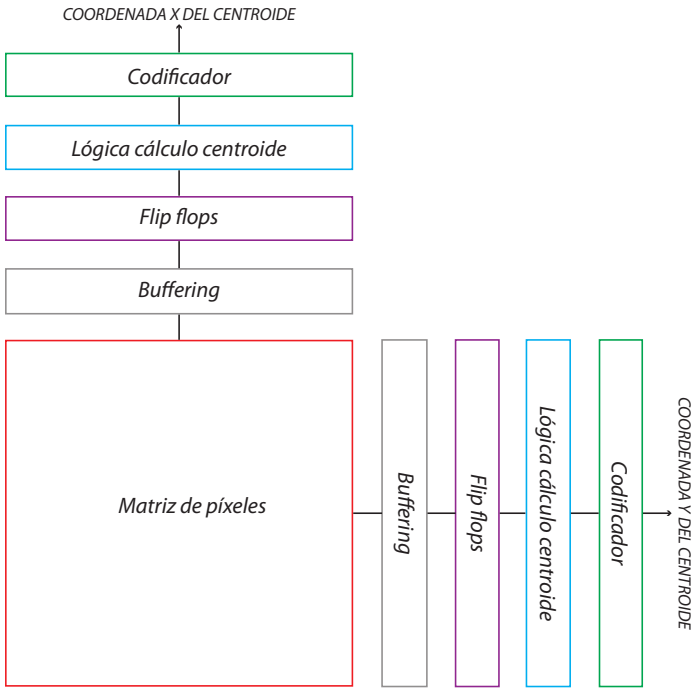

Figura 4.8 Esquema de la arquitectura del sensor Continuo, compuesto por la matriz de píxeles y la periferia.

4.2.1 Buffering

En el modo de operación Continuo se dan las mismas condiciones en lo relativo a la transmisión de las señales que las detalladas para el Octopus en la sección 4.1.1, por lo que el diseño que se da como solución es el mismo. La única diferencia radicará en que en este caso la matriz de píxeles sólo comunica con la periferia mediante las señales de petición, que son señales de salida, por lo que sería necesario solo un bloque de buffers. (Uno por fila/columna). Podemos ver una representación esquemática del mismo en la Figura 4.9.

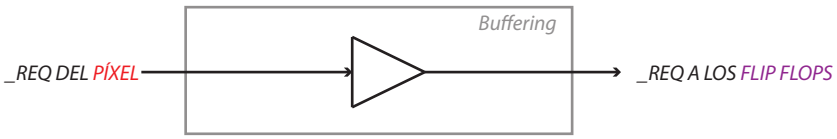

Figura 4.9 Esquema del bloque de buffering de la periferia del sensor Continuo.

4.2.2 Flip Flops

Dado que el dato de salida (coordenadas del centroide) puede ser requerido por el dispositivo externo en cualquier momento, que no tiene por qué tener sincronía en absoluto con los cambios de actitud (que es lo que genera cambios en la salida), se podrían dar glitches en la lógica operacional debidos a condiciones de carrera. Para evitar esto se incorpora una etapa al procesamiento de la señal en la periferia, compuesta por tantos Flip-Flops como filas/columnas haya, que se encargarán de generar un vector de salida de datos sobre cuya velocidad de lectura tenga control la periferia, que generará dicha velocidad en función a la señal de *enable* que actuará como señal de reloj de un biestable convencional.

4.2.3 Bloques de lógica de cálculo del centroide

Es en este bloque en el que encontramos la lógica circuital encargada de proporcionar las coordenadas del centroide, lo cual se lleva a cabo computando el centro de las líneas activadas. La lógica que se implementa con este fin está compuesta por varias etapas en cascada, y cada una de estas etapas está compuesta a su vez por un filtro de bordes y un multiplexor. Vemos una representación de esto en la Figura 4.10 Así, el filtro de bordes elimina en cada etapa las dos líneas contiguas a una desactivada, de forma que para N entradas necesitaremos $N/2$ etapas para asegurarnos de que el filtrado sea suficiente. Esto se consigue implementando un operador AND entre cada dos líneas vecinas. La función del multiplexor es que si en alguna etapa anterior a la última ya se han filtrado todas las líneas y llegado al resultado, no se pierda esta información ante la salida negativa del filtro de bordes. De esta forma, si todas las salidas del filtro de borde son negativas, se copiará la salida de la etapa anterior como salida de ésta. Además de las $N/2$ etapas necesarias en un principio, se incorpora una última etapa de filtrado puesto que existe la posibilidad de que si un número par de líneas esta activa inicialmente, resulten como salida de la última etapa dos líneas. Para determinar una respuesta única como dato que se dará de coordenada del centroide, se implementa una última etapa en la que se da prioridad siempre a la línea superior de entrada. Esto queda ilustrado en el comportamiento del circuito en el Caso 2 del Capítulo 5 de este Trabajo, en el que se llevan a cabo simulaciones sobre el sistema. Podemos por tanto concluir que la resolución de esta lógica de cálculo será de píxel, no llegando a alcanzar la resolución subpíxel. Una forma de alcanzar una resolución de medio píxel sería, mediante una bandera de si esta última etapa ha tenido que actuar o no, determinar si la línea que da como salida es exacta o en realidad la salida sería la coordenada de esa línea sustrayéndole 0.5 píxeles.

Como última característica a tener en cuenta para comprender el funcionamiento y diseño de la lógica de cálculo del centroide, se establece que las líneas exteriores (las perimetrales) nunca tengan preferencia. Es decir, estas son descartadas en la primera etapa, a menos que no haya ninguna otra activa.

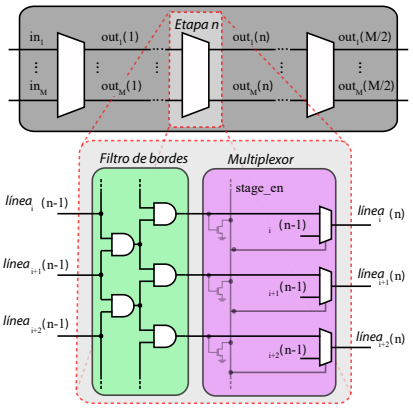

Figura 4.10 Esquema de una etapa del bloque de lógica de cálculo del centroide de la periferia del sensor Continuo.

Finalmente, encontramos en la Figura 4.11 un esquema ilustrativo de cómo se van eliminando las líneas a lo largo de las etapas de cálculo de la coordenada del centroide. En él podemos observar que en caso de haber alguna línea espuria activada, los filtros de borde la eliminan para el cómputo final. Esto será útil cuando el albedo genere dos regiones iluminadas, computando el bloque digital sólo el centroide de la ROI que presente un tamaño mayor [27].

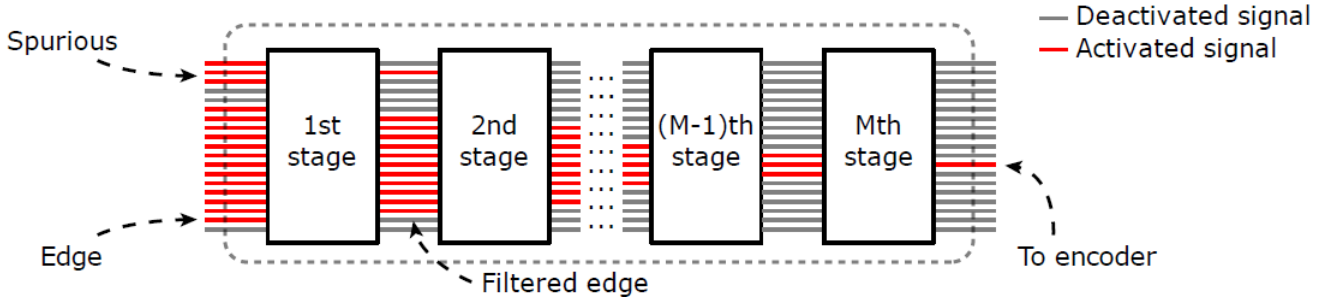

Figura 4.11 Esquema ilustrativo del funcionamiento de la lógica de cálculo del centroide de la periferia del sensor Continuo.

4.2.4 Codificador

La estructura del bloque codificador será la misma que la del sensor Octopus, con la diferencia de que las señales de entrada serán las líneas de salida (tanto negadas como no) de la etapa lógica de cálculo del centroide, de las cuales estará activa la que en ese momento sea la coordenada de salida. De esta forma, un *encoder* genérico sería exactamente igual que el visto en la Figura 4.6, y un ejemplo de un codificador para una matriz 4x4 del sensor Continuo sería el que encontramos a continuación en la Figura 4.12.

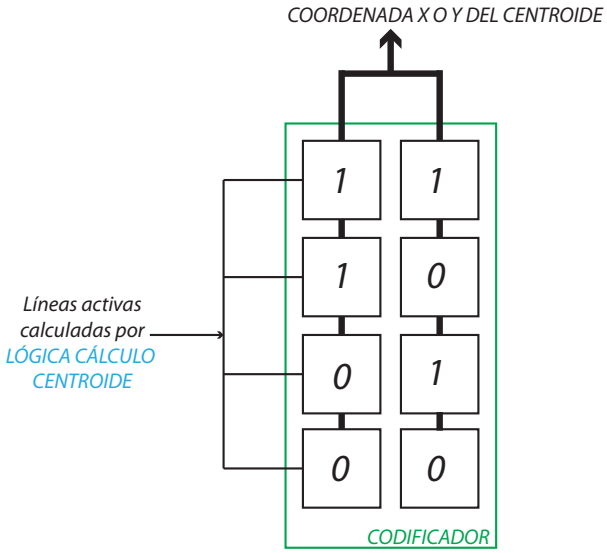

Figura 4.12 Ejemplo del esquema de un codificador para cuatro entradas en la periferia del Sensor Continuo.

5 Simulaciones y resultados

En este capítulo se presentan los resultados obtenidos en el proceso de testeo del funcionamiento de la arquitectura del píxel propuesta anteriormente, el cual se realizó empleando un software de trabajo que cubriese todas las necesidades de la tecnología con la que se trabajó. Por último, se incluye una comparativa entre el diseño realizado y algunos de los sensores existentes mencionados anteriormente en este trabajo.

Con el fin de la verificación del funcionamiento, se utilizó el software computacional para diseño y automatización del diseño electrónico *Cadence*. En esta herramienta se implementó el *Test Bench* (es decir, un entorno en el que verificar la precisión y validez del diseño) que incluyese una matriz que incorporase el píxel diseñado y las periferias necesarias para su lectura. Para ello, lo idóneo sería simular una matriz de 128×128 píxeles (para poder observar un desarrollo lo más parecido posible a lo que ocurrirá en la implementación real), pero esto no resulta práctico, puesto que genera una carga computacional tan elevada que el programa emplearía mucho tiempo y memoria en poder simular la situación. Es por ello que en este trabajo lo que se llevó a cabo fue una simulación de una matriz 4×4 sujeta a estímulos, lo cual nos vale cómo muestra de una matriz 128×128 en la que el resto de los píxeles se encontrasen inactivos. Esta matriz estaría conectada por sus señales de salida y entrada a los periféricos explicados en el Capítulo 4, además de a las correspondientes fuentes de alimentación y señales de control.

Para llevar a cabo las simulaciones, se empleó la siguiente configuración del circuito simulado: se conectó la tensión V_{ph} correspondiente a cada fotodiodo a un modelo de éstos compuesto por una fuente de intensidad y una capacidad en paralelo que representa C_{ph}. La capacidad que presente el fotodiodo no puede ser determinada con total certeza, pero a partir del resultado proporcionado por un extractor de parásitos y de la relación I_{ph}/C_{ph} se estima su valor en un rango de 15 fF. La intensidad de la fuente, que modela la corriente fotogenerada, será prácticamente nula cuando consideremos que no se encuentra iluminado (~ 10 fA) y tomará un valor entre 100 fA y 10 nA cuando este se encuentre generando fotocorriente. (Esta intensidad dependerá del nivel de iluminación y la tensión en la capacidad de integración). Para modelar la impedancia parásita de las líneas de petición, se incorporaron capacidades de 1 pF y resistencias de 640 Ω por cada una de estas líneas, que serían aproximadamente equivalentes a las que encontraríamos en un array de 128×128 píxeles.

Con el objetivo de modelar el periférico de lectura en el modo TFS, se conectó la señal de petición de lectura que realiza la periferia al bus de datos a la señal de confirmación de carga de datos (*ACK* del bus del dispositivo externo que recibe el *sender*), pero generando un retraso de 200 *ns* que simule el tiempo de procesamiento de la solicitud, para lo que se supuso que este dispositivo resolvería la petición en dos ciclos de reloj si trabajase a 10 MHz. La petición de lectura de la periferia al bus de datos se generará con una puerta AND que tiene como entradas las REQ al bus del dispositivo externo que envían los *sender* de filas y columnas.

Una vez se tiene implementado el sistema a simular, el verificar realmente el posible funcionamiento del sistema ante situaciones reales no es un cometido sencillo, y aún más en sistemas que, como el propuesto en este Trabajo Fin de Grado, tratan con estímulos externos (la luz en nuestro caso) cuyo comportamiento no es 100 % predecible. Es por ello que se intentaron plantear las situaciones más extremas, así como las más desfavorables, a la hora de comprobar la funcionalidad de nuestro diseño. A tal efecto se plantean tres posibilidades:

1. Que todos los píxeles se encuentren muy iluminados, lo cual causaría la emisión simultánea de peticiones de lectura.
2. Que todos los píxeles se encuentren expuestos a un nivel medio de intensidad luminosa.
3. Que encontremos píxeles a distintos niveles de iluminación además de píxeles inactivos.

Una vez se hayan realizado estos análisis para testar la funcionalidad, se llevará a cabo un análisis de *Corner*s, cuyo objetivo es comprobar si el diseño funcionaría en las situaciones más extremas de las características de fabricación del chip que pueden darse. Por último, se someterá al sistema a un análisis de Monte Carlo. Los circuitos electrónicos están formados por componentes electrónicos simples, pero debido a los procesos de fabricación presentan un comportamiento estadístico; así, mediante este análisis estudiaremos estadísticamente la funcionalidad del dispositivo en el supuesto de que se diera *mismatching* entre las propiedades de respuesta de los componentes (fundamentalmente de los transistores).

5.1 Verificación funcional

Cómo ya se introdujo anteriormente, para verificar la funcionalidad del diseño estudiaremos los tres escenarios propuestos. Se representarán las señales más relevantes del protocolo de comunicación, dado que ya se ha expuesto y detallado el funcionamiento de cada una de las partes que compondrán la matriz y su periferia. Por cada caso, se presentan los resultados de cada modo de operación y se comentan los aspectos más relevantes para la comprensión de éstos, teniéndose en cuenta todo lo explicado respecto al asunto con anterioridad en este Trabajo Fin de Grado. Así, las señales que serán mostradas variarán según el modo de funcionamiento, puesto que el protocolo de comunicación con la periferia es distinto y la salida de esta también, y son:

- En el modo Octopus, V_PH, $LOCK$ (sólo en el modo TFS) y $SPIKE$ de cada píxel, $\_REQ\_ROW$, ACK_ROW, $RESET\_ROW$, $\_REQ\_COL$ y $RESET\_COL$ de cada fila y cada columna, y el REQ_BUS y BUS_ACK del bus de datos compartido con el dispositivo externo.
- En el modo Continuo, V_PH, y $SPIKE$ de cada píxel, $\_REQ\_ROW$ y $\_REQ\_COL$ de cada fila y cada columna y BUS_BIT_Y y BUS_BIT_X, que contendrán la información calculada por la periferia sobre la dirección del centroide según su fila y su columna, respectivamente.

En ambos modos se representarán las señales externas de control $RESET$ y PIX_ON para mostrar cuándo se fuerza el reseteo y cuando se inicia el periodo de medida. Recordemos que estas dos señales se coordinan según se indicó en la sección 3.3.

A la hora de llevar a cabo la verificación del funcionamiento del diseño, se testó el comparador con un rango amplio de valores de tensión umbral a la entrada, con el objetivo de comprobar su funcionamiento para diversas casuísticas. En las simulaciones que van de la Figura 5.6 a la Figura 5.18 se estableció un voltaje umbral de 750 mV, en el centro del rango de voltaje de entrada. Para la ilustración del tiempo de pulso frente a intensidad fotogenerada, que se presenta en el apartado 5.1.3 se escoge un voltaje umbral de 250 mV, para que en este resultado los tiempos de disparo sean más similares al comportamiento del sensor en la realidad. Esto será así puesto que el diodo realmente no suele superar los 500 mV de tensión en su ánodo, por lo que cuando implementemos el diseño para su aplicación final el voltaje umbral debe ser más bajo que este valor.

5.1.1 Caso 1: Iluminación homogénea y muy elevada

Se configuró la matriz de píxeles de forma que el modelo de los fotodiodos suministrase una corriente de 10 nA (máxima alcanzable) a todos los píxeles por igual. Así, este escenario implica la necesidad de procesar 16 solicitudes (recordemos que trabajamos con una matriz 4×4) totalmente simultáneas en el tiempo, y nos servirá para evaluar el impacto que tiene en la medida el retraso introducido por la propia periferia como consecuencia de la necesidad de arbitración para la transmisión de datos. Este retraso, es decir, el tiempo empleado en procesar cada petición, implicará que a pesar de encontrarse todos los píxeles sometidos al mismo nivel de iluminación, los datos de salida serán distintos, pues cada píxel transmitirá sus datos cuando se le conceda la prioridad desde la periferia. A la hora de analizar los resultados de este escenario, hay que tener en cuenta que es altamente improbable que esta situación se de. Cuando un haz de luz impacte en la matriz del sensor, generará una *ROI* que quizá tome este tamaño, pero nunca incidirá en todas los píxeles la

misma intensidad luminosa, de forma que este colapso no se daría. Se incluye esta casuística para ilustrar el comportamiento de la periferia a la hora de procesar muchas señales simultáneas, otorgar prioridades y desactivar señales.

Modo Octopus

Comentando en primer lugar el modo Octopus, encontramos que, tal y cómo se esperaba, las peticiones son procesadas secuencialmente según la prioridad de arbitración. Este escenario nos sirve para ilustrar cómo el arbitrador otorga la prioridad en sucesos de igualdad, ya que en este caso todos los píxeles emiten la petición al mismo tiempo. Los arbitradores empleados fueron arbitradores *greedy*, configurándose el protocolo de petición para que se diera prioridad a la línea (ya sea fila o columna) contigua a la respondida frente a todas las demás; vemos que esto es lo que ocurre. Una vez resuelve la petición de una línea, atiende inmediatamente la de su contigua, es decir, la que comparte con ella arbitrador en la primera etapa del bloque. Esto tendrá consecuencias sobre todo a nivel de fila; si hay muchas peticiones simultáneas, una vez atienda una petición de una fila, no pasará a otra hasta que haya atendido todas las restantes activas en esa primera, independientemente de si llegaron antes o después que las demás de la matriz.

Vemos en la Figura 5.6 y la Figura 5.8 los resultados de la simulación de este escenario funcionando en modo TFS y FR. Es destacable el hecho de que para tan alta intensidad, en el modo FR no se pueden procesar todas las peticiones correctamente. Dado que el periodo de SPIKE (es decir, la frecuencia de pulso consecuencia de la intensidad luminosa) es menor que el periodo requerido por la periferia y el dispositivo externo para procesar todas las solicitudes, habrá muchas de estas que queden desatendidas. Podemos ver que sólo llega a procesar totalmente las peticiones de la fila 0 y la 1, y aún así, falsea sus datos puesto que no las procesa una vez por periodo, si no una vez cada dos pulsos. Atiende precisamente esas filas porque recordemos que en el modo Octopus la periferia da prioridad a las filas inferiores. La consecuencia de esto será que la medida del centroide que se pueda calcular a través del vector de datos de salida no sea verídica, pues lo calculará entorno a las filas 0 y 1 cuando realmente éste se halla centrado en la matriz. Hay que tener en cuenta a la hora de sopesar este hecho que el caso de una fotocorriente de 10 *nA* es el caso límite de funcionamiento, puesto que es la corriente máxima que generarían los fotodiodos. Para demostrar que en unas condiciones más normales, y habituales, de funcionamiento, el modo FR funcionaría satisfactoriamente, se presentan el caso 2 y el caso 3 de estas simulaciones. En la presentación de los resultados funcionando en este modo y escenario, se incluyen sólo los 20 primeros microsegundos, para que se puedan apreciar mejor los detalles, puesto que en realidad, desde el primer pulso en adelante el comportamiento se repite periodicaménte; por cómo se configuraron los arbitradores, repite la misma secuencia de respuesta cada cuatro pulsos. Es uno de estos "periodos" los que se amplian en la Figura 5.7, en la que encontramos una ampliación en el eje temporal dónde se indican las señales de cada píxel en particular.

Modo Continuo

En el caso del modo Continuo de funcionamiento, este escenario sirve para ilustrar cómo se ha configurado el algortimo de cálculo del centroide ante sucesos de igualdad. Es decir, vemos reflejado en la salida qué prioridad se ha otorgado a cada línea en la lógica de cálculo. Tal y cómo se planteaba en el capítulo 4, en el modo Continuo se da prioridad a la línea superior y se descartan las líneas perimetrales, lo cual vemos confirmado en que estando las 4 activas, la salida sea $binario : 10 \rightarrow decimal : 2$. Los resultados de esta simulación los encontramos en la Figura 5.10

5.1.2 Caso 2: Iluminación homogénea y de intensidad media

En este escenario se configuró el modelo de fotodiodo para suministrar a todos los píxeles una intensidad de 1 *nA*. En estas condiciones, también todas las solicitudes serán simultáneas, pero se realizarán con menos latencia que en el Caso 1 funcionando en FR, lo cual nos permitirá observar mejor su funcionamiento, tal y como se planteaba anteriormente. Para el resto de modos no se simulará, puesto que los resultados serían muy similares al Caso 1 pero con frecuencias menores, y por tanto no extraeríamos de ellos ninguna conclusión nueva.

Se presentan los resultados obtenidos en la Figura 5.11, donde podemos observar que en este caso sí se procesan todas las peticiones. El error absoluto que obtendremos en la medida será el mismo que en el escenario anterior en TFS, ya que el procesamiento de 16 pulsos simultáneos es idéntico independientemente

de la intensidad luminosa (lo decisivo es el número de peticiones, no el instante temporal de estas). Si encontraríamos un cambio en el error relativo, ya que en este caso, al ser el periodo de oscilación mayor, el impacto del error absoluto es menor.

5.1.3 Caso 3: Iluminación desigual

En estas circunstancias, no todos los píxeles de la matriz se encontrarían iluminados con la misma intensidad. Con el fin de que la prueba aportase resultados interesantes, se decidió dar los niveles de iluminación representados en el esquema de la Figura 5.1. Así, esta distribución podría ser una representación más cercana a la realidad de cuando el sistema se encuentre iluminado por un haz de luz que haya penetrado por el agujero de la óptica. Podremos observar por tanto la evolución de la arbitración en el tiempo y ver claramente la proporcionalidad de la frecuencia de pulso con el nivel de iluminación.

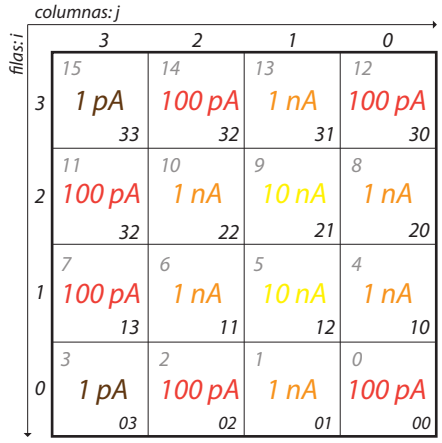

Figura 5.1 Esquemático de la fotocorriente asignada a cada píxel para la simulación en el caso 3, en el que la iluminación es heterogénea. Encontramos cada píxel referenciado según sus coordenadas de la forma (fila,columna), que aparecen como ij en la esquina inferior derecha, o según su número de píxel, que aparece como n en la esquina superior izquierda.

Modo Octopus

En este caso, observamos mucho mejor en el modo Octopus la proporcionalidad de la frecuencia de pulso con la intensidad luminosa.

Funcionando en modo TFS, habrá un solo pulso de cada píxel, y podemos ver como en efecto se comprueba el hecho planteado con anterioridad en este trabajo de que el píxel más iluminado y el que primero emite el pulso suele coincidir con el centroide del haz de luz. Para esta distribución de intensidades, el centroide se hallaría en la fila 1, columna 1.5, y el píxel que primero ve su petición atendida es el 5, cuyas coordenadas son fila 1, columna 1. Vemos así que en la última etapa de arbitración los arbitradores *greedy* dan prioridad a la línea inferior (justo al contrario que la lógica de cálculo en el Continuo). Esto se trata simplemente de una característica del diseño y podría cambiarse de forma sencilla.

Observamos los resultados que se darían en este escenario en la Figura 5.12. Encontramos dentro de los 50 μs de simulación dos momentos relevantes: el primero, entorno a los 4 μs, en el que emiten el pulso los píxeles 5 y 9, tal y como se esperaba, por ser los más iluminados con una fotocorriente asignada de 10 nA; un segundo, entorno a los 22 μs en el que los píxeles con una iluminación de 1 nA hacen sus peticiones. Podemos concluir también de esta simulación que píxeles iluminados con menor intensidad, cómo son los restantes de la distribución planteada, necesitan un tiempo superior al empleado para aportar resultados, pues el periodo de su pulso es mayor que el tiempo de medida. En este caso sólo obtendríamos resultados de los píxeles con una corriente superior a 1 nA, dado que se tomó un periodo de medida de 50 μA. Para ilustrar cómo afecta el nivel de iluminación, y en consecuencia la magnitud de la corriente generada en el fotodiodo, al tiempo que tarda en dar un pulso el comparador, se incluye la Figura 5.2. En ella encontramos

una representación del tiempo en escala logarítmica frente al voltaje a la salida del comparador (señal *spike*). De esta simulación podemos extraer la relación de datos del tiempo que tarda el comparador en emitir un pulso en función de la fotocorriente que se de en el diodo, conjunto que se representa en la Figura 5.3.

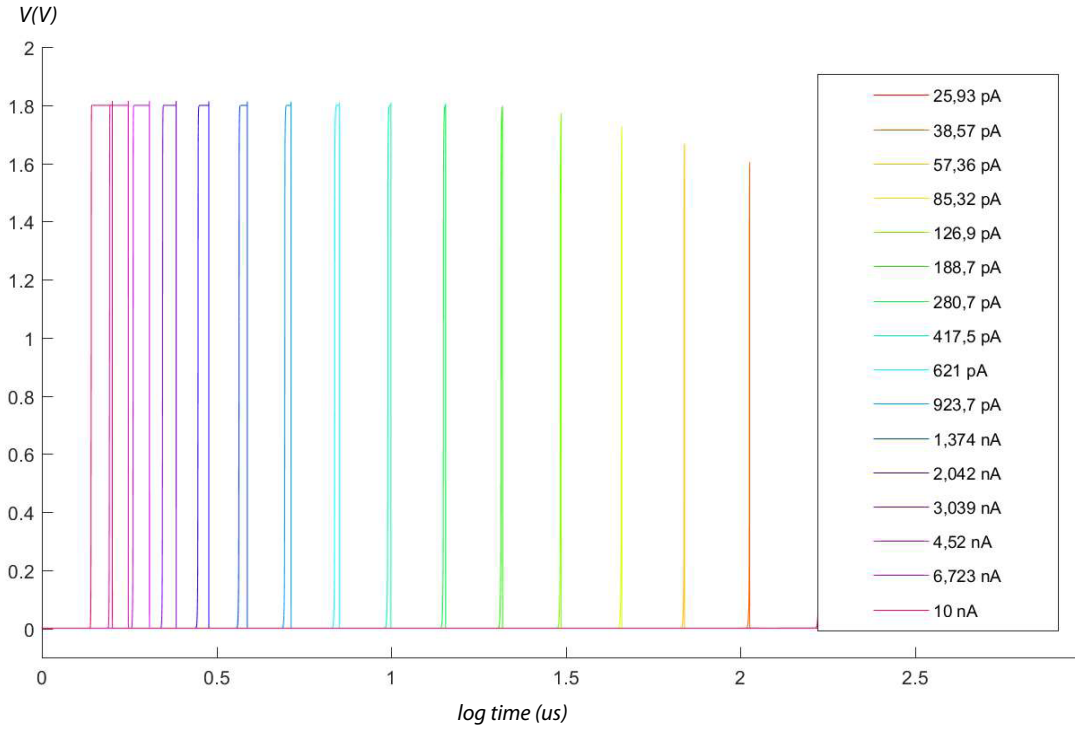

Figura 5.2 Representación V frente a t de la señal *spike* de un píxel según la intensidad de la fotocorriente (indicada en la leyenda) con el objetivo de ilustrar la variación del tiempo transcurrido hasta el primer pulso. El eje temporal se encuentra en formato logarítmico. Se incluyen sólo aquellas que emiten este pulso antes de los 300 μs.

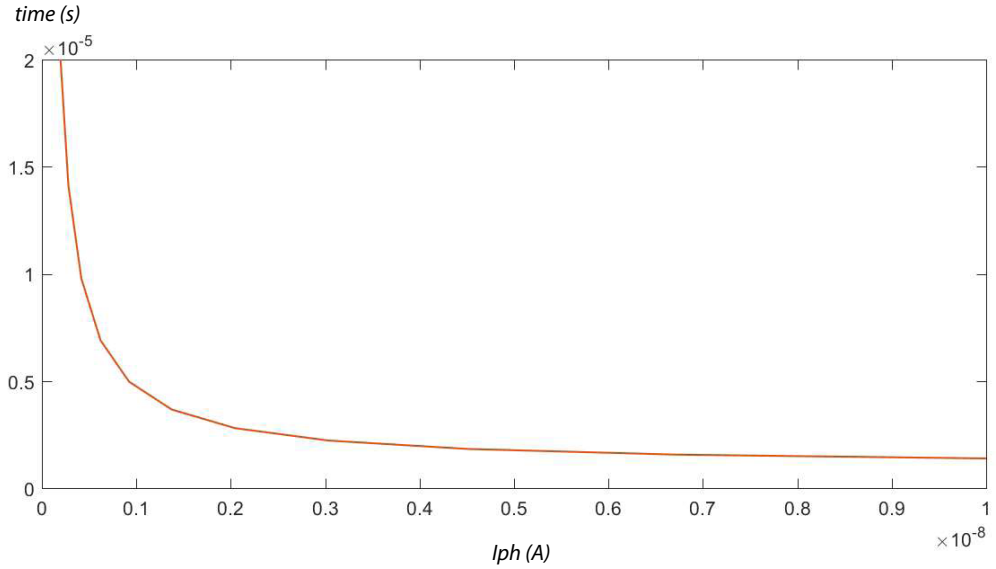

Figura 5.3 Representación t_{spike} versus I_{ph} .

Funcionando en modo FR (Figura 5.13), vemos una ilustración perfecta de la variedad de frecuencias en función del nivel de iluminación. Podemos observar a simple vista que se dan dos frecuencias de pulso; la de los dos píxeles más iluminados, y la de los siguientes seis (son los mismos dos subgrupos que en TFS). Así, la situación más desfavorable es la que se da en los instantes señalados en la representación, en los que colapsan las peticiones de los dos subgrupos en la periferia. En el tiempo que toma procesar todas las solicitudes del subgrupo de seis píxeles con fotocorriente de 1 *nA*, los iluminados a 100 *nA* emiten dos pulsos más que no serán procesados correctamente. Aún así, podemos apreciar que para este modo de funcionamiento el error cometido es mucho menor que en los escenarios anteriores, puesto que no llegan a colapsar tantas peticiones.

Para observar con claridad qué peticiones han sido emitidas y atendidas si nos encontramos en el caso FR, lo más sencillo es observar V_{ph}, dónde veremos con claridad qué petición ha sido emitida, y *RESET_PIX* para ver cuándo ha sido aceptada cada petición, puesto que esta señal se activará en cada píxel cuando se haya completado su protocolo de comunicación con la periferia.

Modo Continuo

En este escenario en el modo Continuo, observamos cómo se van activando las peticiones progresivamente en el tiempo, pero al hallarse distribuidas de forma relativamente simétrica (que es como ocurriría al incidir la luz a través de un agujero circular) el cómputo del centro al final permanece constante.

La representación de los resultados se presenta en la Figura 5.14, dónde encontramos señalado un momento en el que encontramos una anomalía en la salida. En el instante en el que los píxeles con una fotocorriente de 1 *nA* alcanzan el voltaje umbral y su comparador emite un pulso, vemos que la información de los buses de datos, particularmente el de la x, presentan una discontinuidad. Esto ocurre porque debido a la propia implementación de las líneas, sus impedancias... las señales no son procesadas de forma totalmente instantánea, lo que provoca que en ese pequeño retraso la salida pueda presentar fluctuaciones. Como vemos, inmediatamente después se vuelve al resultado anterior, puesto que una vez recibidas todas las peticiones el cálculo del centroide mediante la lógica de la periferia seguirá dando el mismo resultado, tal y como se ha comentado en este Trabajo Fin de Grado.

5.2 Análisis de *Corners*

El análisis de *corners* es una técnica de experimentación sobre diseños electrónicos que se basa en comprobar el funcionamiento del sistema en cuestión ante la variación de los parámetros de fabricación del circuito integrado. En concreto, se analizará la variación de la velocidad de respuesta y transición de estados de los dos tipos de transistores que encontramos: PMOS y NMOS. Así, el análisis recibe este nombre por plantear las cuatros "esquinas", o situaciones límite, dentro de las posibles combinaciones de casuísticas. Estas situaciones serían las que se ilustran en la Figura 5.4.

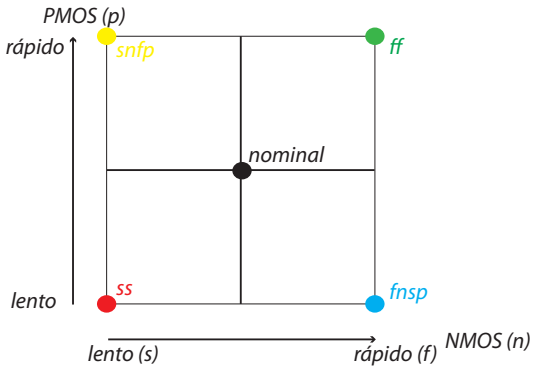

Figura 5.4 Representación de las cuatro situaciones límites (*corners*) en lo relativo a la velocidad de respuesta de cada tipo de transistor en un circuito integrado. $ss = slow\ slow$, $snfp = slow N\ fast P$, $fnsp = fast N\ slow P$, $ff = fast\ fast$.

Para realizar este análisis al sistema, se empleó una configuración simple, con el único objetivo de verificar cómo influían las variaciones en la fabricación en un píxel activo y si producía que se generasen pulsos espúreos en aquellos no iluminados. Así, para hacer las simulaciones lo más ligeras computacionalmente posibles, se planteó que un píxel de la matriz 4×4 estuviera iluminado y el resto no. Se probó el resultado para los tres modos de funcionamiento. Encontramos estos resultados expuestos en la Figura 5.15, Figura 5.16 y Figura 5.17. Vemos que el diseño funciona para los tres en los cuatro *corners*, y podemos por tanto asegurar que, mientras se fabrique dentro de las tolerancias establecidas, el píxel cumplirá con su funcionalidad.

Tal y cómo se deseaba, no se producen pulsos espúreos en los píxeles no iluminados. Observamos sin embargo, y como era de esperar, una influencia de la velocidad de respuesta de los transistores en el tiempo que tarda en emitir un pulso el comparador. Podemos apreciar fácilmente, y se comprobó en los datos experimentales, que el tiempo del primer pulso será el mismo en los tres modos de funcionamiento. Esto es lo lógico dado que el comparador es el mismo en los tres casos y en el primer pulso no influye la realimentación, que sí dependería de cada modo. Así, podemos calcular sencillamente cuánto influye el *corner* en el que nos encontremos en el tiempo del primer pulso respecto del nominal, sabiendo que este dato será independiente de si el píxel funciona en modo Octopus o Continuo.

Para calcular esto, aplicaremos la fórmula 5.1 para el cómputo del error relativo, teniendo en cuenta que la medida para estas simulaciones (al igual que para todas las anteriores de este estudio) se ha iniciado en el instante $t = 1\ \mu s$, por lo que referenciaremos todos los tiempos como $t = t_{absoluto} - 1\ \mu s$.

$$\varepsilon = \frac{|t_{pulso\,corner} - t_{pulso\,nominal}|}{t_{pulso\,nominal}} \tag{5.1}$$

Y por tanto, calculándolo para cada *corner*, si tenemos en cuenta que $t_{pulso\,nominal} = 6{,}462\ \mu s$:

Corner ff: fast fast

$$t_{pulso\,ff} = 6{,}5325\ \mu s$$

$$\varepsilon_{ff} = 0{,}0108\ \rightarrow 1{,}08\,\% \tag{5.2}$$

Corner fnsp: fast N slow P

$$t_{pulso\,fnsp} = 6{,}5375\ \mu s$$

$$\varepsilon_{fnsp} = 0{,}0116\ \rightarrow 1{,}16\,\% \tag{5.3}$$

Corner snfp: slow N fast P

$$t_{pulso\,snfp} = 6{,}6560\ \mu s$$

$$\varepsilon_{snfp} = 0{,}0299\ \rightarrow 2{,}99\,\% \tag{5.4}$$

Corner ss: slow slow

$$t_{pulso\,ss} = 6{,}5278\ \mu s$$

$$\varepsilon_{ss} = \rightarrow 1{,}01\,\% \tag{5.5}$$

5.3 Análisis de Monte Carlo

A la hora de realizar el diseño del circuito para el píxel, hemos tenido en cuenta parámetros aproximados caracterizados por métodos deterministas que se basan en la física de cada componente. Éstos nos han permitido realizar un diseño del dispositivo y una predicción de su funcionamiento. Ya en el apartado anterior, estudiamos qué ocurriría en cada situación limítrofe dentro de las posibles características de fabricación, que causarían una variación en el comportamiento de ciertos componentes respecto del predecido.

En este apartado nos dedicaremos a aplicar el método estadístico denominado Método de Monte Carlo al píxel, que consiste en asignar a cada variable que demos cómo entrada (en este caso la velocidad de respuesta de los transistores, característica de su fabricación) un valor dentro de los correspondientes a esta variable modelada según una distribución gaussiana, tomando dentro de esa distribución un valor aleatorio distinto en cada muestra. Así, manejando acertadamente las variaciones de estas variables podremos obtener un estudio de la fiabilidad de nuestro diseño, así como una estimación de la sensibilidad que presentan ciertas variables de salida. En este caso, como variable de salida, analizaremos el *SPIKE* del comparador y el tiempo que tarda en dar su primer pulso, al igual que en los apartados anteriores.

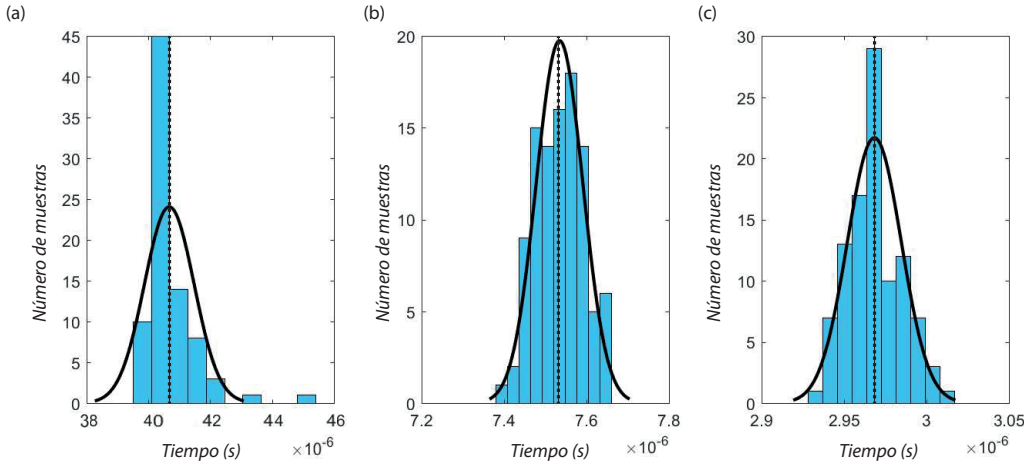

Figura 5.5 Análisis Monte Carlo del tiempo transcurrido hasta el primer pulso a la salida del comparador. Histograma con la distribución de las cien muestras. (a) $I_{ph} = 500\ pA$. (b) $I_{ph} = 3\ nA$. (c) $I_{ph} = 10\ nA$.

En la Figura 5.18 se presentan las señales *SPIKE* de salida, al aplicar un análisis Monte Carlo , de cuatro configuraciones de píxel distintas: las tres primeras, son píxeles iluminados a diferentes intensidades, y la última sería un píxel no iluminado. Este último se incluye para asegurarnos de que no se producirán pulsos espúreos. Las tres señales que contienen información se incluyen también ampliadas para poder observar mejor la distribución resultante de las distintas muestras tomadas en el análisis.

En concreto, el análisis que se realizó fue un análisis de Monte Carlo con 100 muestras sobre píxeles generando una corriente de 10 *nA*, 3 *nA*, 500 *pA* y 10 *pA* (píxel inactivo estático). Sobre este, se analizó el tiempo que tardaba el comparador en emitir el primer pulso. Los histogramas obtenidos se han representado en la Figura 5.5 y los valores medios ($\overline{\mu}$), la desviación típica absoluta (σ) y la relativa ($\overline{\mu}/\sigma$), se incluyen en la Tabla 5.1.

Tabla 5.1 Análisis Monte Carlo del tiempo hasta el primer pulso.

I_{ph}	$\overline{\mu}$	σ	$\overline{\mu}/\sigma(\%)$
500 *pA*	40,67 μs	0,80 μs	1,97 %
3 *nA*	7,53 μs	0,06 μs	0,75 %
10 *nA*	2,97 μs	0,02 μs	0,55 %

Como ya podíamos apreciar a simple vista en la Figura 5.18, cuánto menor es la corriente fotogenerada, más afecta el mismatching entre transistores al funcionamiento del comparador. No obstante, podemos concluir que en general la fiabilidad del diseño es elevada y las variables son moderadamente sensibles al *mismatching*.

5.4 Benchmarking

Las especificaciones del sensor propuesto, estimadas mediante simulación y resultados previos, han sido resumidas y comparadas con algunos de los sensores ya existentes en la Tabla 5.2. En ella se presentan las características que presentan el sensor Octopus, el sensor Octopus en modo TFS [5] y sensor Continuo [4], el sensor *Galileo* desarrollado por la *ESA* [13] y el desarrollado por Xie et al. [15], además de las obtenidas para nuestro diseño con la simulación. Esta comparativa servirá además como ilustración de las ventajas de las arquitecturas elegidas para combinar frente a otras existentes. La caracterización del sensor Octopus (FR) fue llevada a cabo en el laboratorio del Instituto de Microelectrónica de Sevilla por el grupo TIC-179 de la US.

Tabla 5.2 Especificaciones de los sensores de partida y especificaciones resultantes en el sensor diseñado.

Sensor	Sensor Octopus	Sensor Octopus TFS [5]	Sensor Continuo [4]	Galileo ESA [13]	Xie et al. [15]	Este trabajo
Tipo	Sensor de luminosidad octopus (a)	Sensor de luminosidad octopus basado en eventos (b)	Sensor de luminosidad de modo continuo (c)	Sensor digital APS	Sensor digital APS	(a), (b), (c) en función del modo
Tecnología	UMC 0.18 μm	AMS UMC 0.18 μm HV	UMC 0.18 μm	UMC 0.18 μm	UMC 0.18 μm 1P4M	UMC 0.18 μm
Tensión de alimentación	1,8 V	5,8/ 3,3 V	1,8 V	3,3/ 1,8 V	3,3/ 1,8 V	1,8 V
Cantidad de datos	Coordenadas de píxeles iluminados (a)	1-100 eventos (b)	Coordenadas centroide (x_c,y_c) (c)	1 frame/ROI	368 pixels/ 25x25=945 pixels	(a), (b), (c) en función del modo
Precisión	0,30º (θ) 1,47º (ϕ) (a)	0,013º (θ) 0,05º (ϕ) (b)	0,78º (θ) 3,33º (ϕ) (c)	0,024º	0,01º	ND
Consumo de potencia	1,07 mW @ 1000 fps [a]	52 mW	63 μW @1000 fps [a]	520 mW	42,73 mW	Octopus: 118 mW, Continuo: 32 mW
Frame rate	5 fps/lux (a)	>200 fps @1klux (b)	500.000 fps (c)	10 fps	10 fps	(a), (b), (c) en función del modo

[a] Bajo unas condiciones de iluminación de 1 klux.

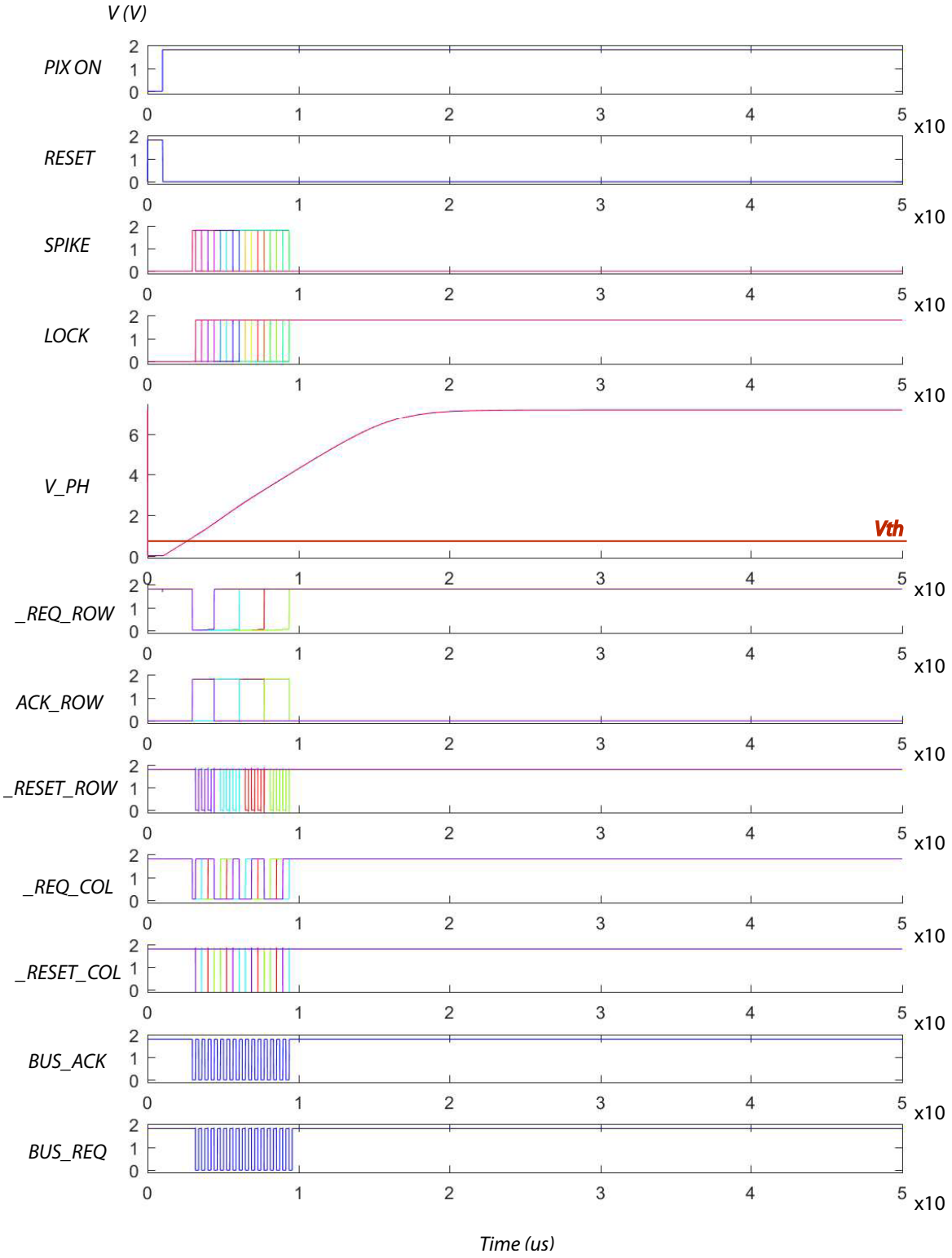

Figura 5.6 Resultados de la simulación en modo Octopus TFS del funcionamiento del sistema expuesto a iluminación homogénea (caso 1) durante 50 μs.

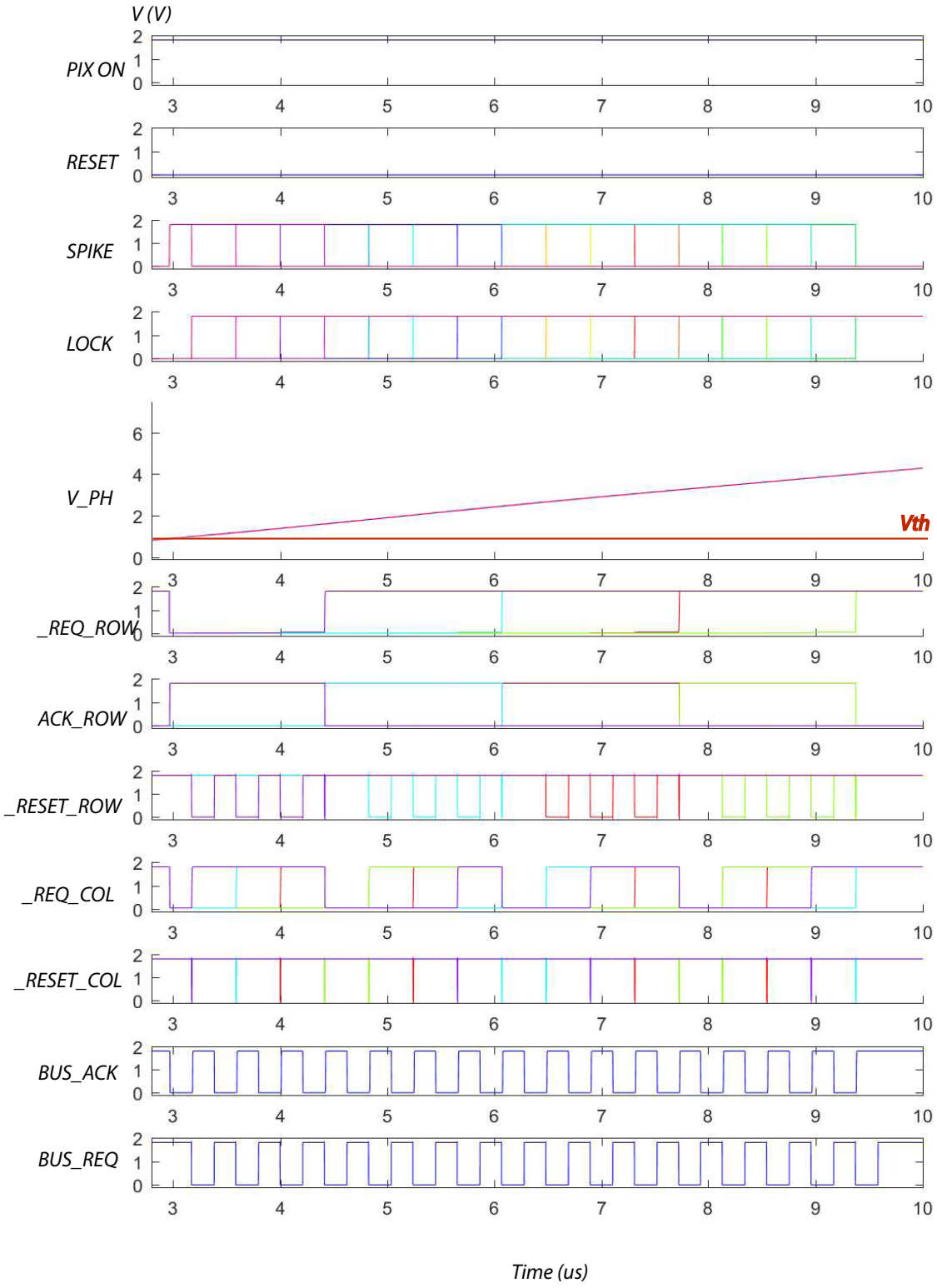

Figura 5.7 Resultados de la simulación en modo Octopus TFS del funcionamiento del sistema expuesto a iluminación homogénea (caso 1) con ampliación en el eje temporal para observar el protocolo de funcionamiento.

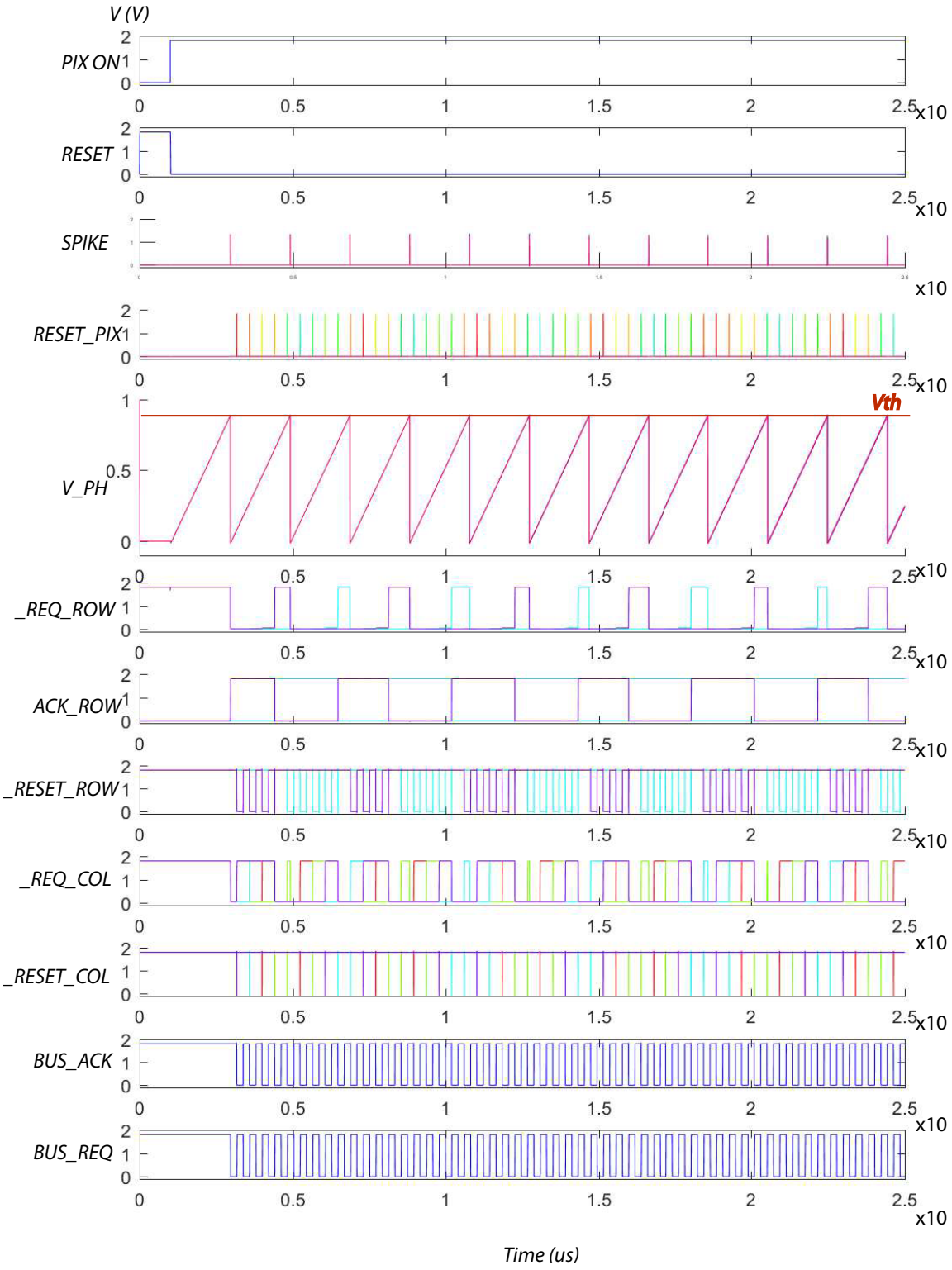

Figura 5.8 Resultados de la simulación en modo Octopus FR del funcionamiento del sistema expuesto a iluminación homogénea (caso 1) durante 25 μs.

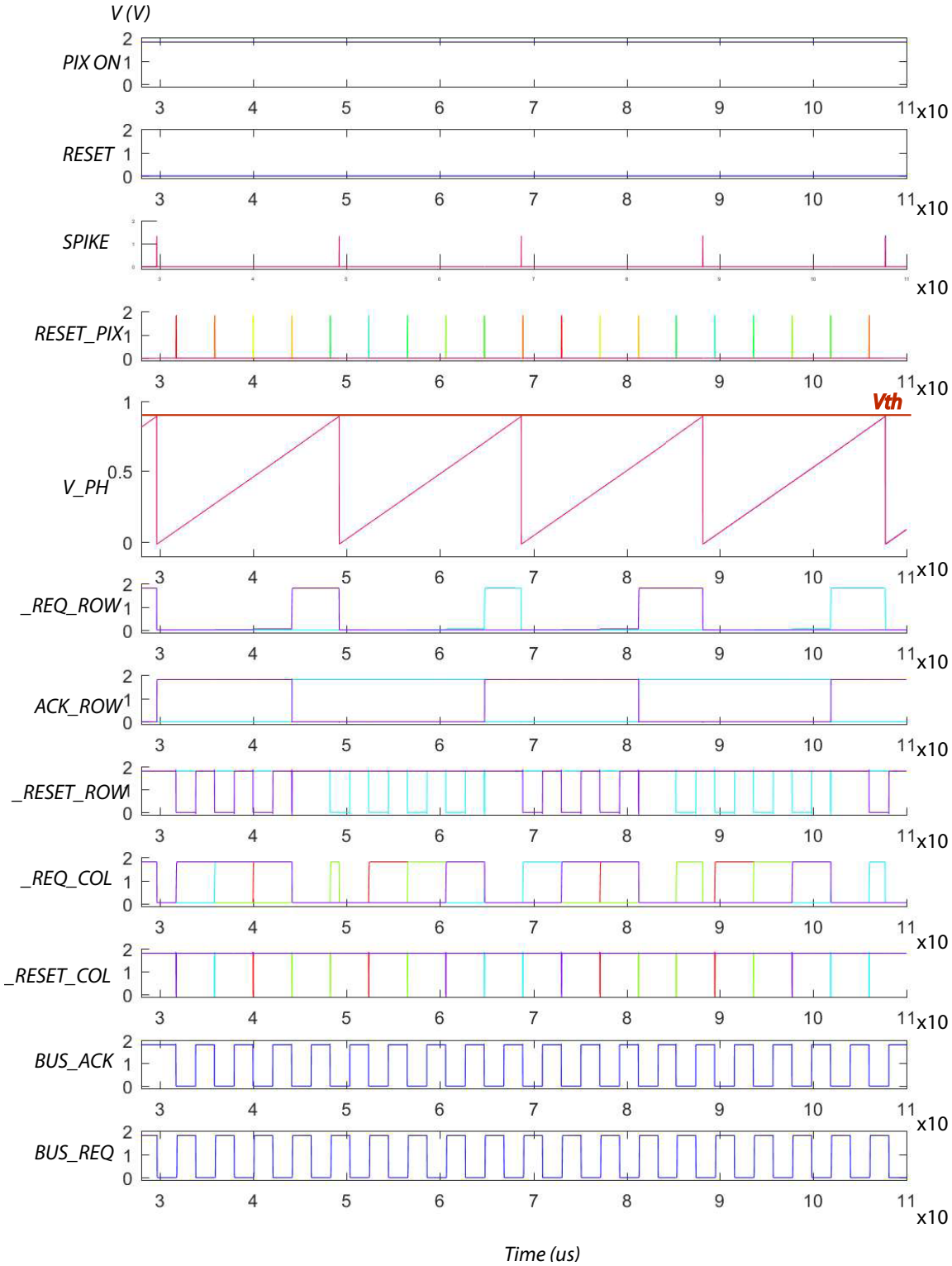

Figura 5.9 Resultados de la simulación en modo Octopus FR del funcionamiento del sistema expuesto a iluminación homogénea (caso 1) con ampliación en el eje temporal para observar el protocolo de funcionamiento.

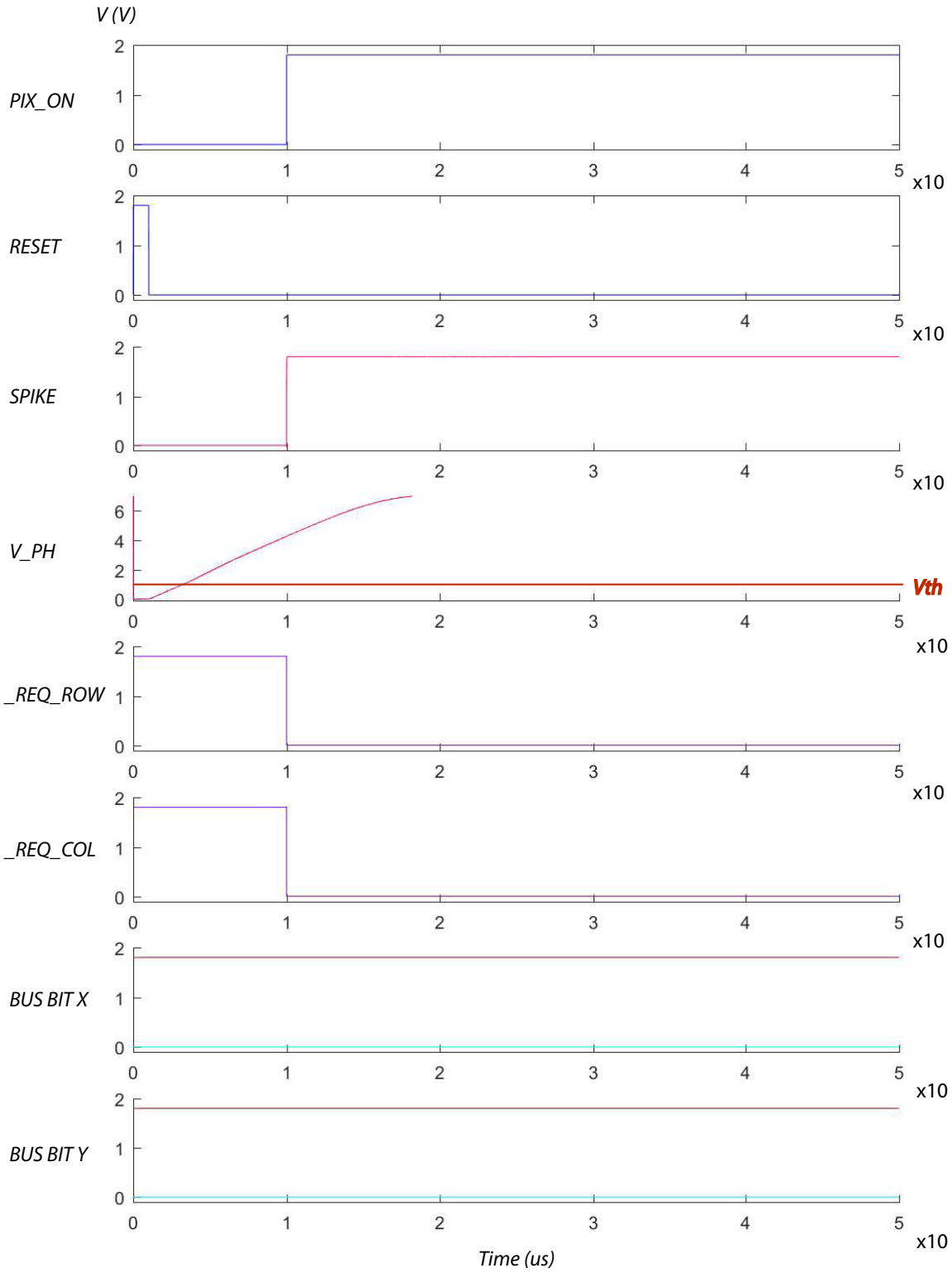

Figura 5.10 Resultados de la simulación en modo Continuo del funcionamiento del sistema expuesto a iluminación homogénea (caso 1) durante 50 μs.

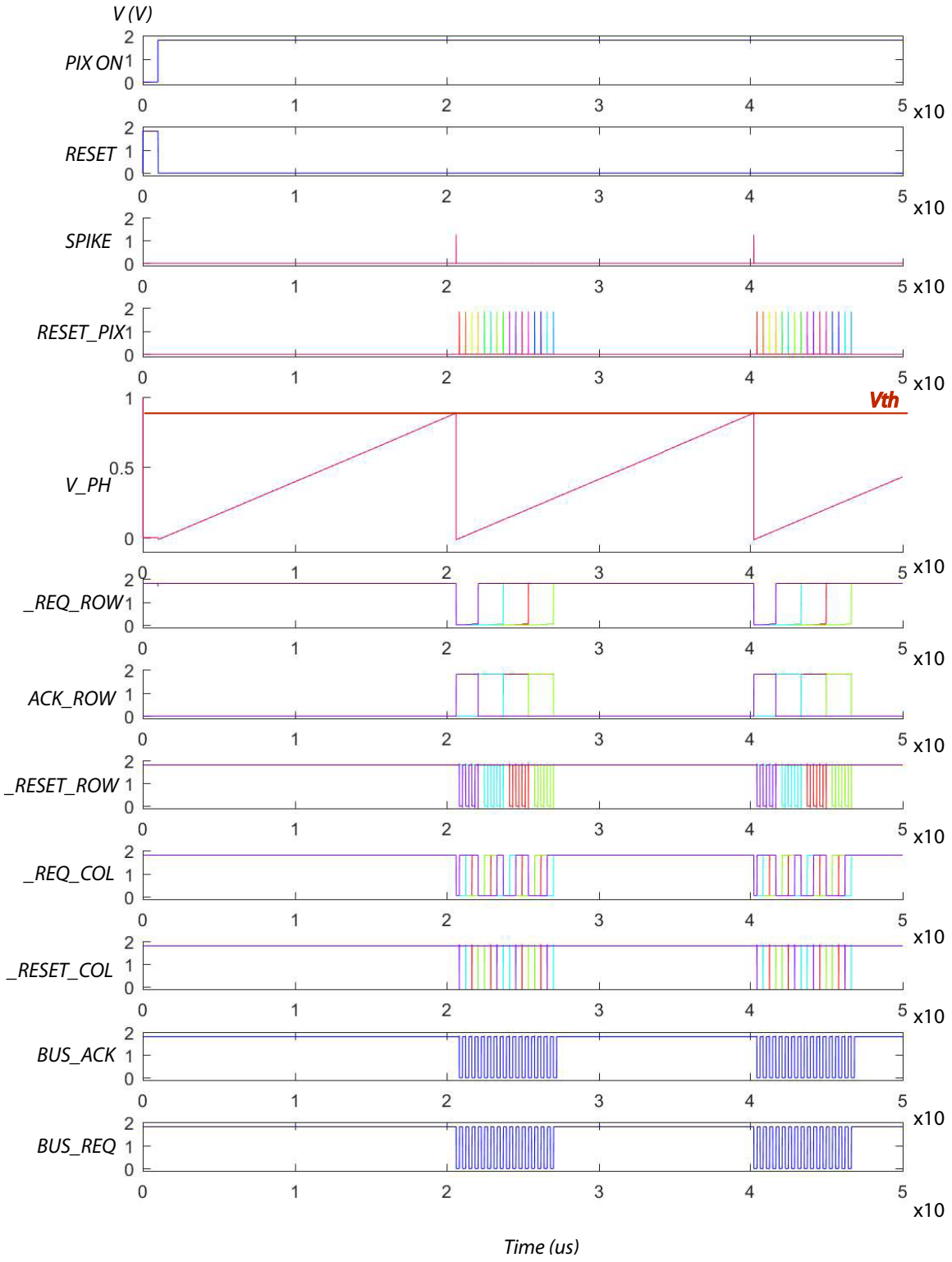

Figura 5.11 Resultados de la simulación en modo Octopus FR del funcionamiento del sistema expuesto a iluminación homogénea (caso 2) durante 50 μs.

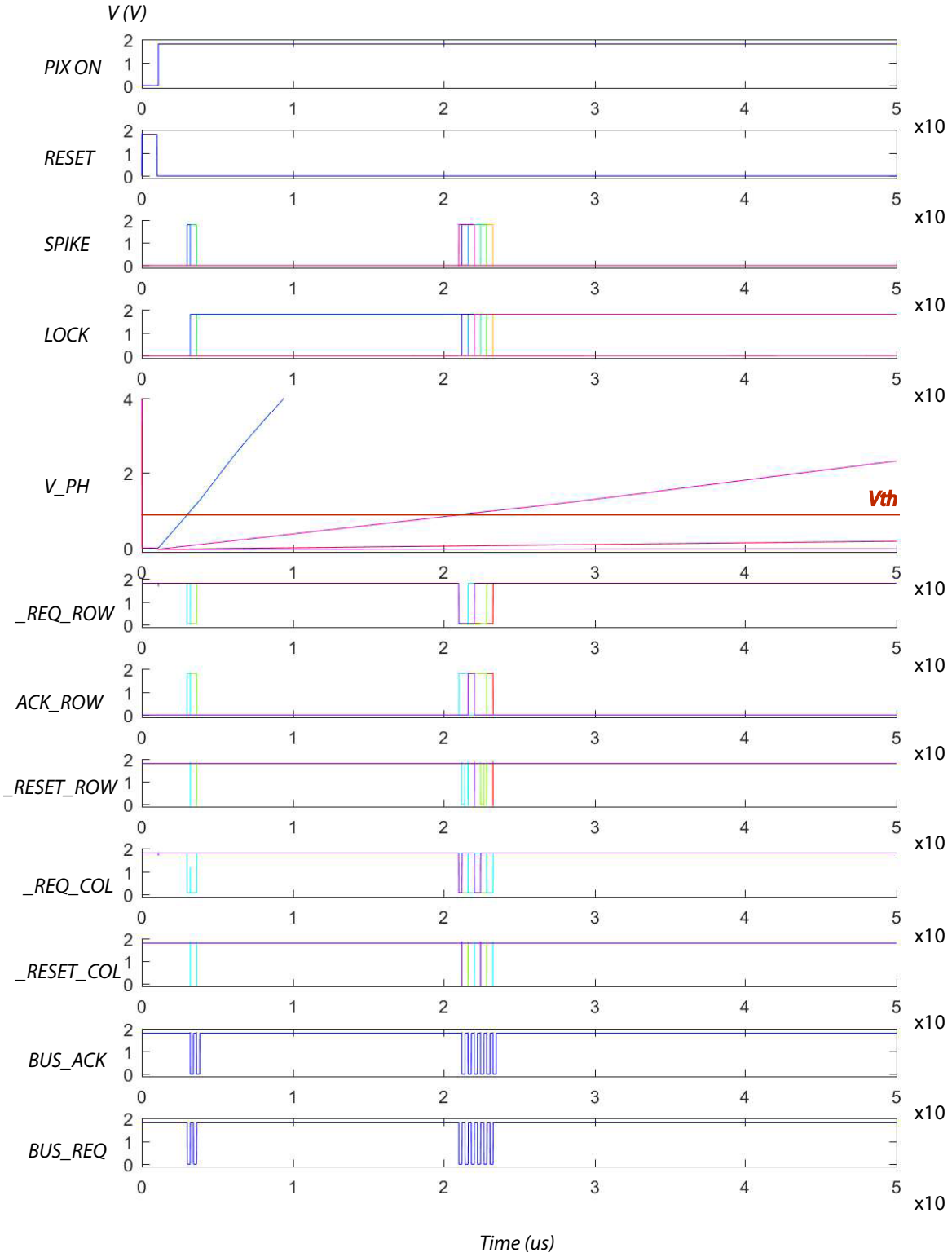

Figura 5.12 Resultados de la simulación en modo Octopus TFS del funcionamiento del sistema expuesto a iluminación desigual (caso 3) durante 50 μs.

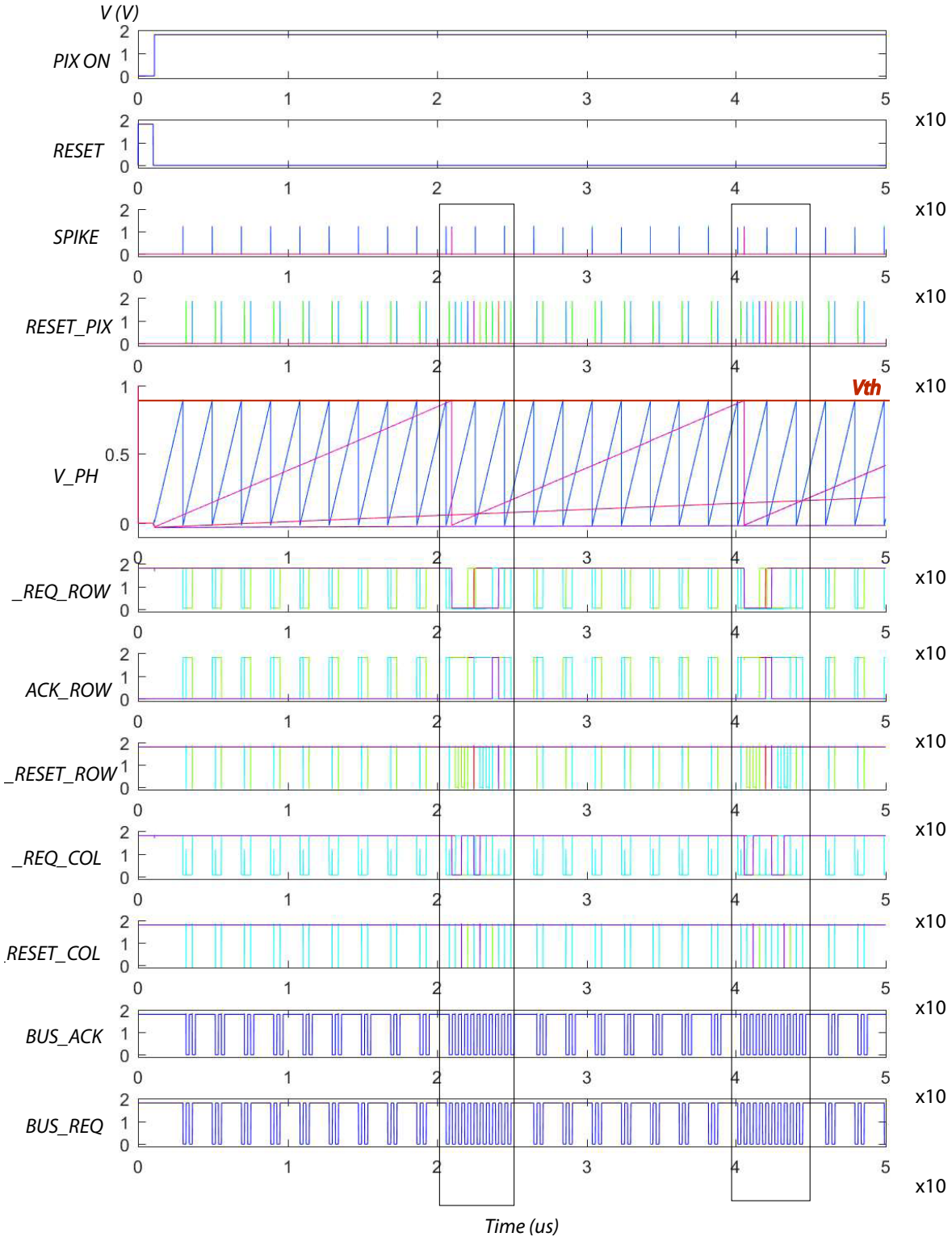

Figura 5.13 Resultados de la simulación en modo Octopus FR del funcionamiento del sistema expuesto a iluminación desigual (caso 3) durante 50us.

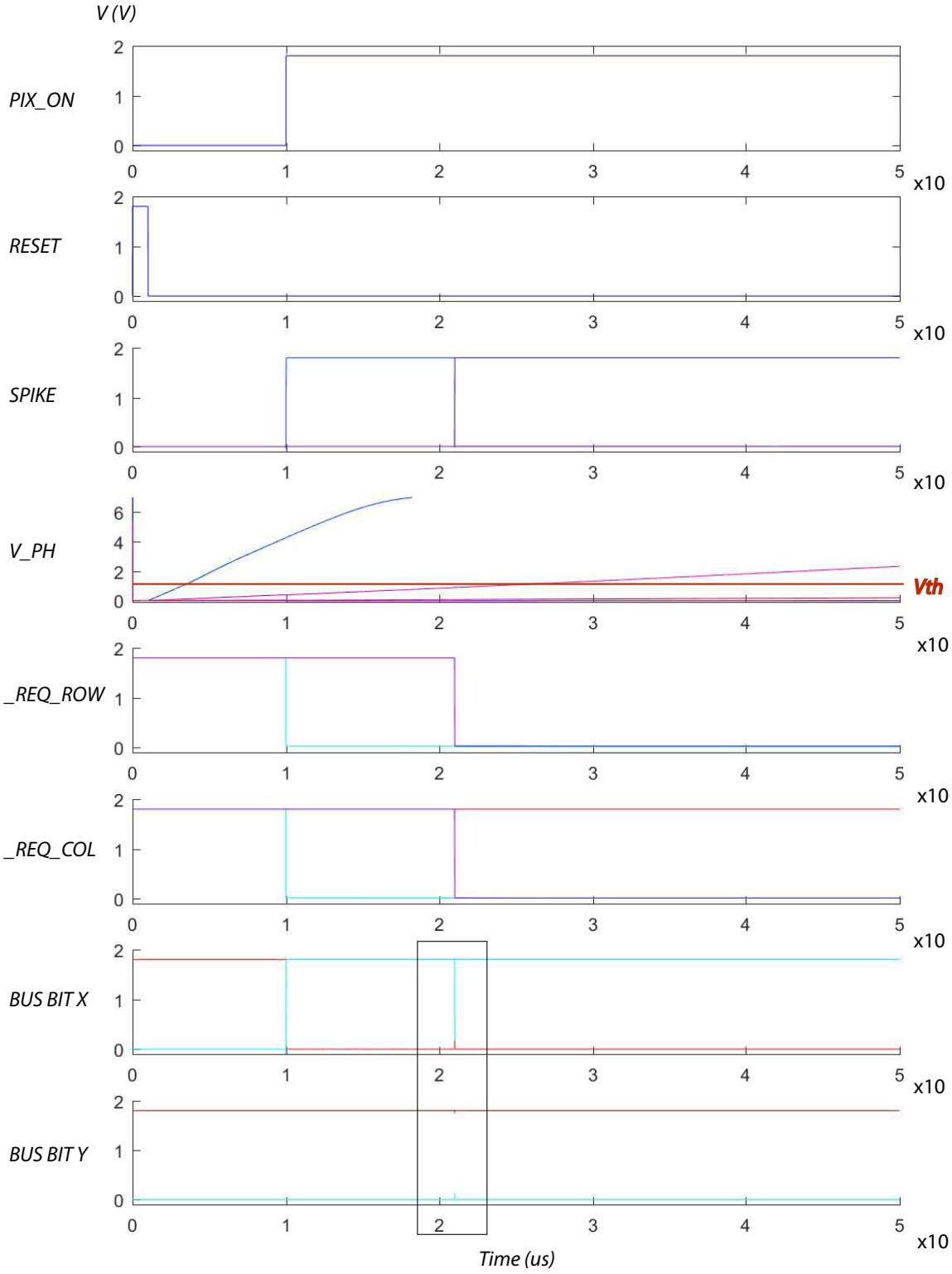

Figura 5.14 Resultados de la simulación en modo Continuo del funcionamiento del sistema expuesto a iluminación desigual (caso 2) durante 50 μs.

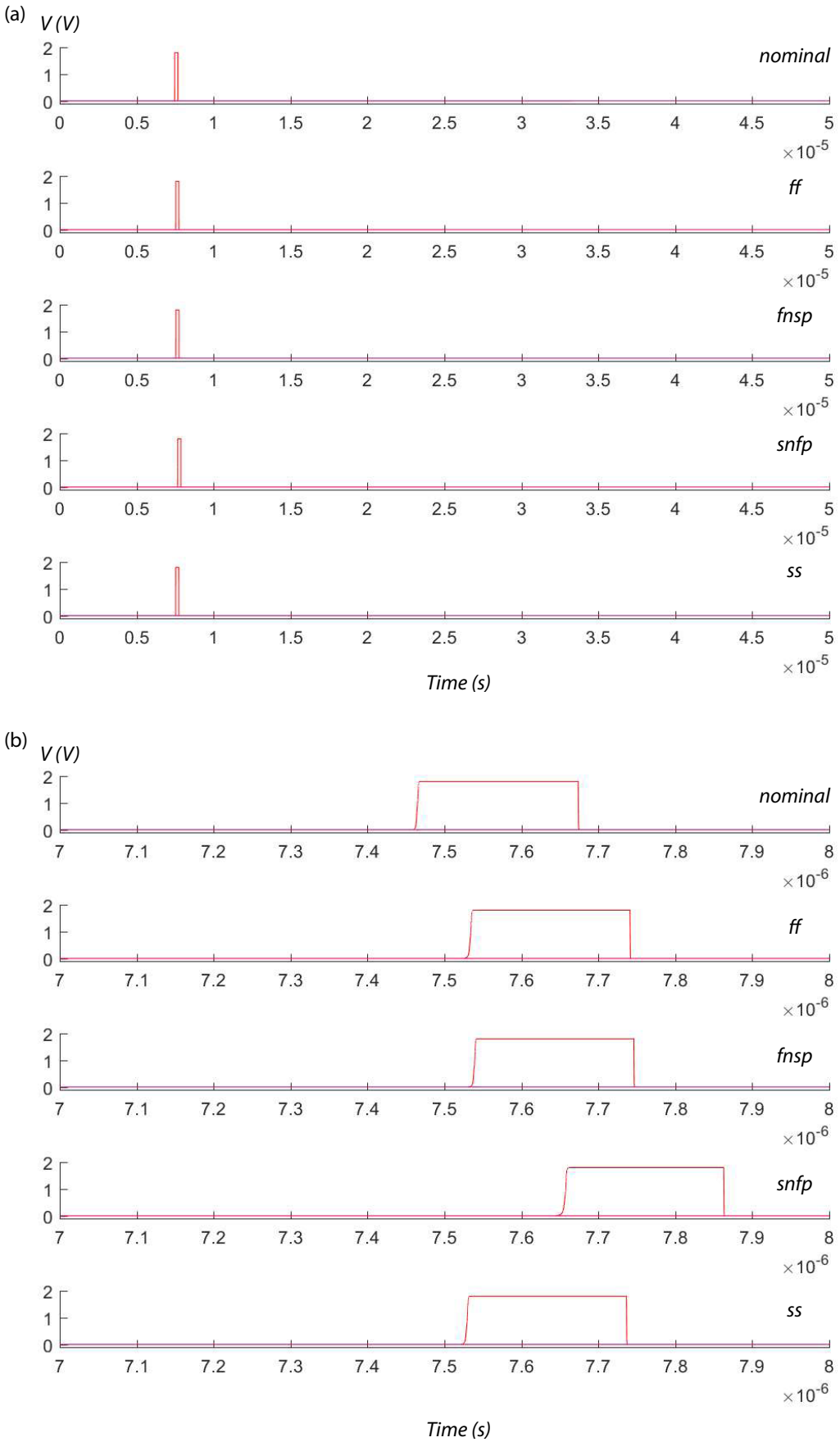

Figura 5.15 Señales *SPIKE* del píxel activo resultado del análisis de *corners* para el sistema funcionando en modo Octopus TFS, con el píxel 15 iluminado con una intensidad de 3 *nA*. (a) Simulación durante 50 μs. (b) Ampliación en el eje temporal del momento del pulso.

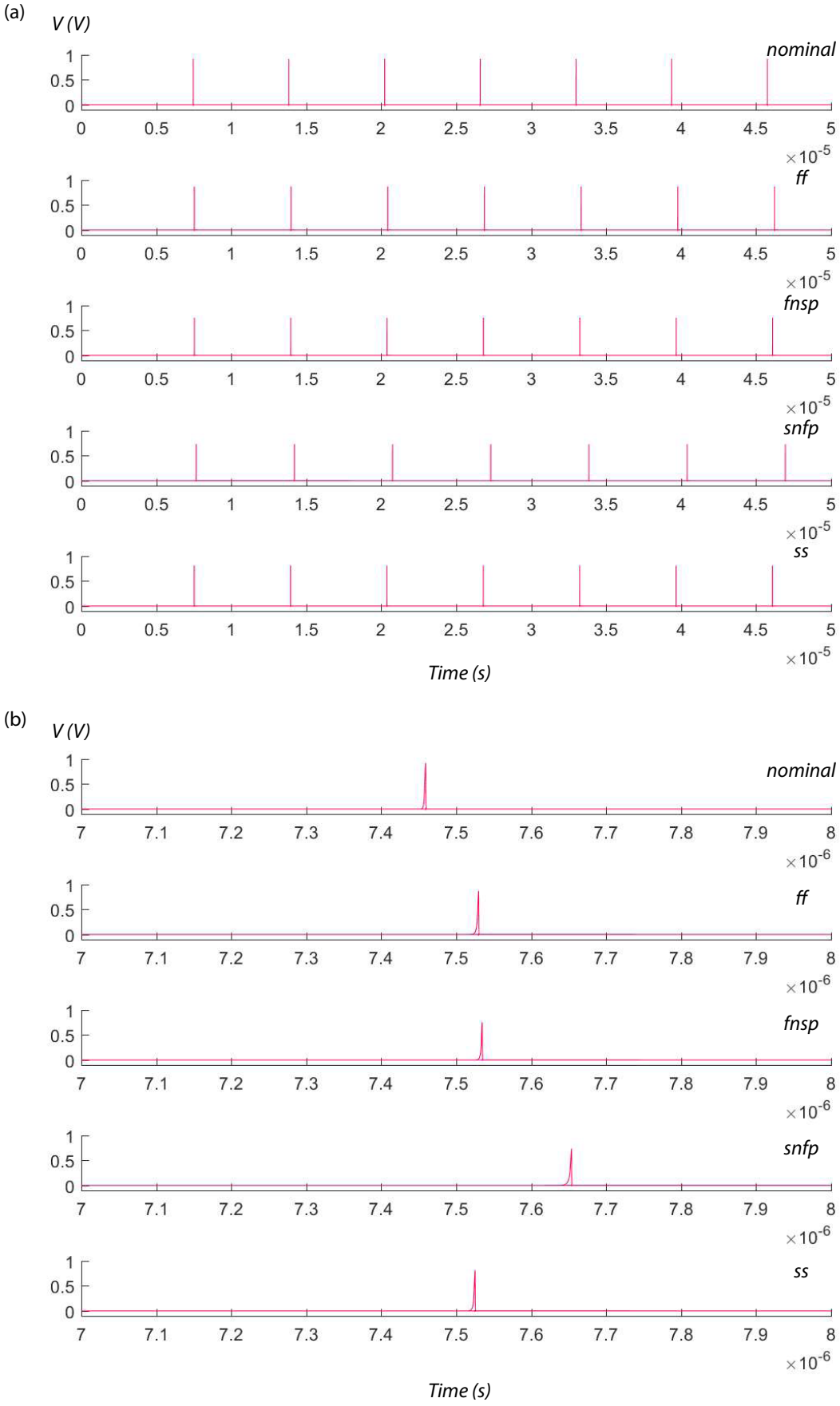

Figura 5.16 Señales *SPIKE* del píxel activo resultado del análisis de *corners* para el sistema funcionando en modo Octopus FR, con el píxel 15 iluminado con una intensidad de 3 nA. (a) Simulación durante 50 μs. (b) Ampliación en el eje temporal del momento del primer pulso.

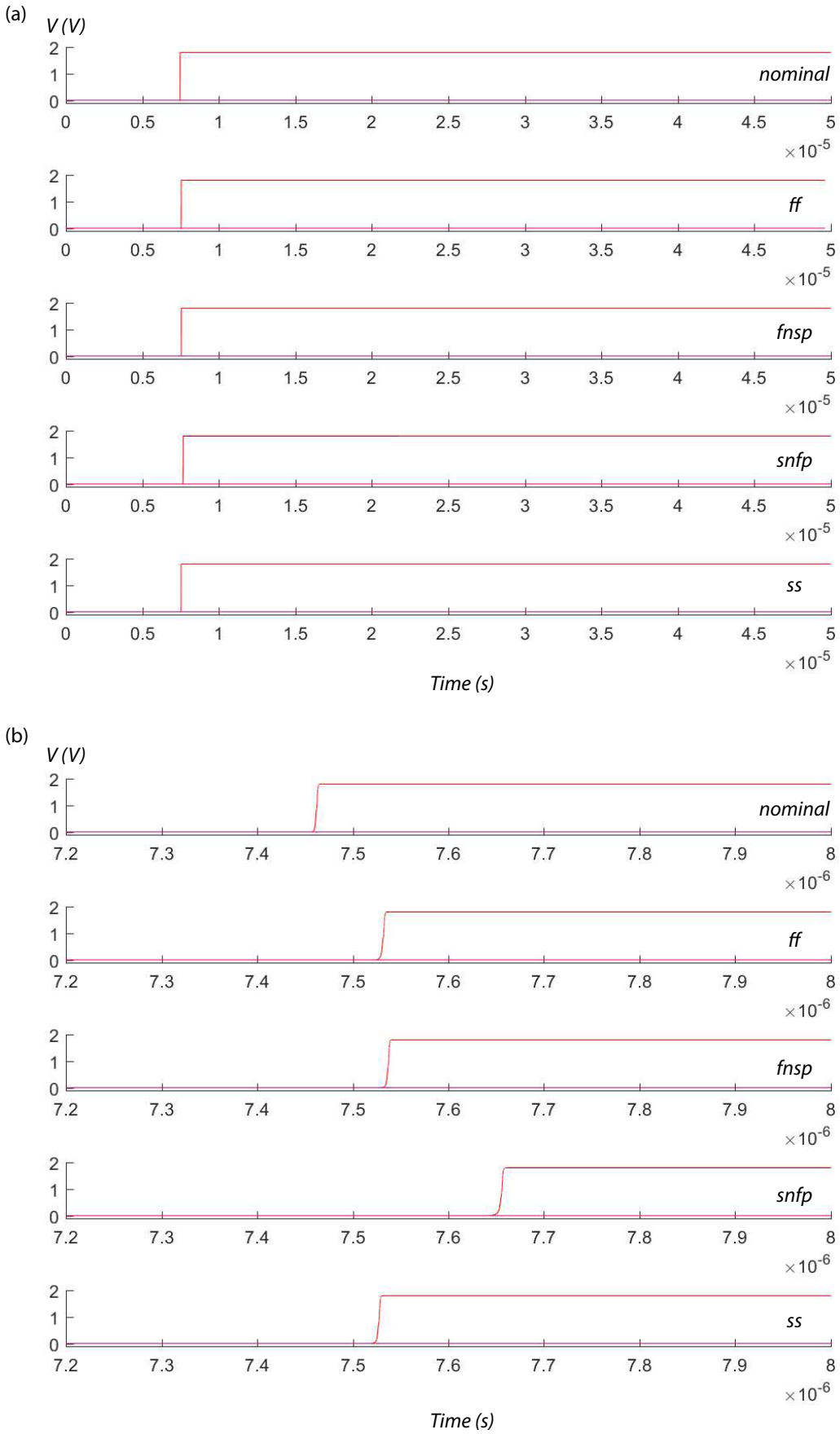

Figura 5.17 Señales *SPIKE* del píxel activo resultado del análisis de *corners* para el sistema funcionando en modo Continuo, con el píxel 15 iluminado con una intensidad de 3 *nA*. (a) Simulación durante 50 μs. (b) Ampliación en el eje temporal del momento del pulso.

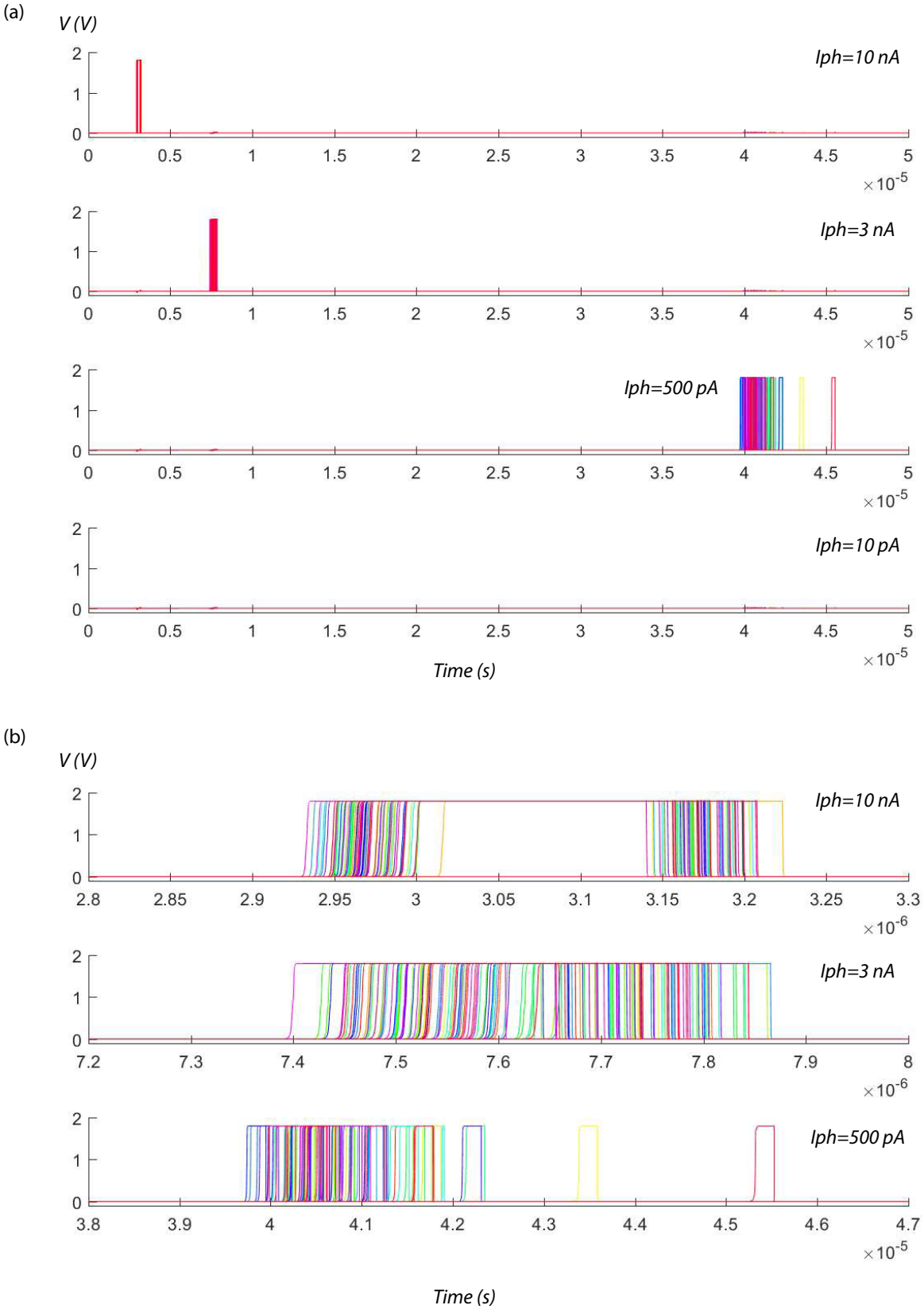

Figura 5.18 Señales *SPIKE* que genera el comparador a su salida para tres píxeles iluminados con diferente intensidad, resultadas de un análisis de Montecarlo con un número de muestras $N = 100$. (a) Simulación durante 50 μs. (b) Ampliación en el eje temporal del momento del pulso para cada intensidad luminosa.

6 Conclusión y trabajo futuro

A lo largo de este estudio se ha diseñado un píxel para sensores de control de la actitud aeroespacial con diferentes modos de funcionamiento, de forma que permitiese elegir entre uno u otro en función de las necesidades a las que tenga que responder el sistema en cada momento. Así, se ha conseguido reunir en un solo diseño las ventajas que presentaba cada modo de operación. La arquitectura Octopus en modo *Free-Running* tiene una alta resolución, con un error de precisión pequeño y un bajo consumo de potencia. La arquitectura Octopus en modo *Time-to-First-Spike* nos proporcionaba también una alta resolución pero con un tiempo de respuesta menor y menor consumo (ya que el comparador de cada píxel se bloquea al emitir el primer pulso), a cambio de un error ligeramente mayor al anterior. La arquitectura continua aporta una mayor sencillez de implementación, al dar como salida directamente las coordenadas del centroide, a costa de la perdida de la resolución subpíxel.

Se ha desarrollado un diseño efectivo en cuanto a funcionalidad y respuesta conforme a unas especificaciones previas obtenidas tras el test de varios prototipos. Para validar el diseño se llevaron a cabo diversas simulaciones, en las que se obtuvieron los resultados esperados.

Incluyendo los aspectos mencionados anteriormente, y las tareas futuras a abordar que surgen como consecuencia de la complejidad de la materia de estudio, podríamos plantear como líneas abiertas:

- La implementación del layout del layout de la matriz de píxeles y de los circuitos periféricos. Debido a la extensión esperada para un estudio de estas características, no se incluyó la realización del layout, quedando esto pendiente cómo trabajo futuro.
- La profundización en el diseño del comparador, especialmente en la disminución del consumo. Para este estudio se empleó un modelo de comparador previamente realizado por el grupo TIC-179, dado que estaba probada su efectividad y eficiencia. Sin embargo, podría estudiarse más exhaustivamente para mejorar este aspecto.
- La implementación de un fotodiodo más eficiente que los empleados hasta el momento.
- El estudio del diseño robusto a radiación (*rad-hard*), que implica el uso de técnicas de diseño y layout avanzadas.
- La implementación *on-chip* de una interfaz de comunicación estándar: *SPI*, *UART*, *I2C*, etc. para el transporte de datos en la comunicación con periféricos externos.

Índice de Figuras

Índice de Tablas

Bibliografía

[1] "Euroconsult report about sun sensors, https://digital-platform.euroconsult-ec.com/product/prospects-for-the-small-satellite-market/."

[2] L. Salgado-Conrado, "A review on sun position sensors used in solar applications," *Renewable and Sustainable Energy Reviews*, vol. 82, pp. 2128–2146, 2018.

[3] J. Lenero-Bardallo, R. Carmona-Galán, and Á. Rodríguez-Vázquez, "A high dynamic range image sensor with linear response based on asynchronous event detection," in *2015 European Conference on Circuit Theory and Design (ECCTD)*. IEEE, 2015, pp. 1–4.

[4] R. Gomez-Merchan, M. López-Carmona, J. Leñero-Bardallo, and Á. Rodríguez-Vázquez, "A high-speed low-power sun sensor with solar cells and continuous operation," in *ESSDERC 2021-IEEE 51st European Solid-State Device Research Conference (ESSDERC)*. IEEE, 2021, pp. 147–150.

[5] J. A. Leñero-Bardallo, L. Farian, J. M. Guerrero-Rodríguez, R. Carmona-Galán, and A. Rodríguez-Vázquez, "Sun sensor based on a luminance spiking pixel array," *IEEE Sensors Journal*, vol. 17, no. 20, pp. 6578–6588, Oct 2017.

[6] L. Farian, P. Häfliger, and J. A. Leñero-Bardallo, "Miniaturized sun sensor with in-pixel processing for attitude determination of micro space probes," in *2015 International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, June 2015, pp. 1–6.

[7] A. Ali and F. Tanveer, "Low-cost design and development of 2-axis digital sun sensor," *Journal of Space Technology*, vol. 1, no. 1, pp. 1–5, June 2011.

[8] P. Ortega, G. Lopez-Rodriguez, J. Ricart, M. Dominguez, L. M. Castaner, J. M. Quero, C. L. Tarrida, J. Garcia, M. Reina, A. Gras, and M. Angulo, "A miniaturized two axis sun sensor for attitude control of nano-satellites," *IEEE Sensors Journal*, vol. 10, no. 10, pp. 1623–1632, 2010.

[9] J. M. Quero, C. Aracil, L. G. Franquelo, J. Ricart, P. R. Ortega, M. Dominguez, L. M. Castaner, and R. Osuna, "Tracking control system using an incident radiation angle microsensor," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 2, pp. 1207–1216, April 2007.

[10] D. Brasoveanu and J. Sedlak, "Analysis of earth albedo effect on sun sensor measurements based on theoretical model and mission experience," *Advances in the Astronautical Sciences*, vol. 100, pp. 485–498, 1998.

[11] G. N. Tiwari and S. Dubey, *Fundamentals of photovoltaic modules and their applications*. Royal Society of Chemistry, 2009.

[12] J. R. Wertz, *Spacecraft Attitude Determination and Control*. D. Reidel Publishing Co., The Netherlands, 1978.

[13] F. Boldrini, E. Monnini, D. Procopio, B. Alison, W. Ogiers, M. Innocent, A. Pritchard, and S. Airey, "Attitude sensors on a chip: Feasibility study and breadboarding activities," in *Procceedings of 32nd Annual AAS Guided Control Conference*, February 2009, pp. 1197–1216.

[14] C. C. Liebe, S. Mobasser, Y. Bae, C. J. Wrigley, J. R. Schroeder, and A. M. Howard, "Micro sun sensor," in *Proceedings, IEEE Aerospace Conference*, vol. 5. IEEE, 2002, pp. 5–5.

[15] N. Xie and A. J. P. Theuwissen, "A miniaturized micro-digital sun sensor by means of low-power low-noise CMOS imager," *IEEE Sensors Journal*, vol. 14, no. 1, pp. 96–103, Jan 2014.

[16] E. R. Fossum, "Active pixel sensors: Are CCDs dinosaurs?" in *Charge-Coupled Devices and Solid State Optical Sensors III*, vol. 1900. International Society for Optics and Photonics, 1993, pp. 2–14.

[17] P. Fortescue, G. Swinerd, and J. Stark, *Spacecraft Systems Engineering*, 4th ed. Wiley, 2011.

[18] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, and T. Delbruck, "Retinomorphic event-based vision sensors: Bioinspired cameras with spiking output," *Proceedings of the IEEE*, vol. 102, no. 10, pp. 1470–1484, Oct 2014.

[19] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629–1636, 1990.

[20] K. Fukushima, Y. Yamaguchi, M. Yasuda, and S. Nagata, "An electronic model of the retina," *Proceedings of the IEEE*, vol. 58, no. 12, pp. 1950–1951, 1970.

[21] K. A. Boahen, "Point-to-point connectivity between neuromorphic chips using address events," *IEEE Transactions on Circuits and Systems II*, vol. 47, no. 5, pp. 416–434, 2000.

[22] A. Mortara, E. A. Vittoz, and P. Venier, "A communication scheme for analog VLSI perceptive systems," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 6, pp. 660–669, 1995.

[23] M. Silvilotti, "Wiring considerations in analog VLSI systems with application to field-programmable networks," Ph.D. dissertation, Cal. Inst. of Tech., Pasadena, California, 1991.

[24] E. Culurciello, R. Etienne-Cummings, and K. A. Boahen, "A biomorphic digital image sensor," *IEEE journal of solid-state circuits*, vol. 38, no. 2, pp. 281–294, 2003.

[25] U. K. Mishra and J. Singh, *Semiconductor device physics and design*. Springer, 2008, vol. 83.

[26] D. G. Chen, D. Matolin, A. Bermak, and C. Posch, "Pulse-modulation imaging—review and performance analysis," *IEEE transactions on biomedical circuits and systems*, vol. 5, no. 1, pp. 64–82, 2011.

[27] P. Appel, "Attitude estimation from magnetometer and earth-albedo-corrected coarse sun sensor measurements," *Acta Astronautica*, vol. 56, no. 1-2, pp. 115–126, 2005.