Revista de I+D TECNOLÓGICO
Portal de revistas: http://revistas.utp.ac.pa

Arquitectura Basada en UART para Sincronización de Dispositivos MEMS: Implementación en FPGA y Análisis ASIC

UART-Based Architecture for MEMS Devices Synchronization: FPGA Implementation and ASIC Analysis

Diego Bouche[1], Edwin Acevedo[1], Maytee Zambrano[1,2], Fernando Arias[1,2], Edson Galagarza[1]

[1]Grupo de Investigación en Tecnologías Avanzadas de Telecomunicación y Procesamiento de Señales (GITTS), Universidad Tecnológica de Panamá, Panamá. [2]Centro de Estudios Multidisciplinarios en Ciencias, Ingeniería y Tecnología AIP (CEMCIT-AIP), Universidad Tecnológica de Panamá, Panamá.

***Autor de correspondencia:** edson.galagarza@utp.ac.pa

RESUMEN. Este trabajo aborda la importancia del control preciso de señales digitales para la sincronización entre un dispositivo MEMS (*Microelectromechanical Systems*) y un espectrómetro perteneciente a un prototipo de laboratorio. La sincronización se implementó mediante una tarjeta FPGA (*Field-Programmable Gate Array*), cuya reconfigurabilidad, alta precisión, reducido tamaño y bajo costo relativo ofrecen ventajas significativas frente a opciones como microcontroladores o sistemas de adquisición de datos (DAQ). La configuración del prototipo se gestiona desde una computadora, y la comunicación con la FPGA se realiza mediante el protocolo UART (*Universal Asynchronous Receiver/Transmitter*) a través de una conexión USB. Dado que las FPGA no cuentan con librerías integradas, fue necesario diseñar desde cero el módulo UART, el cual fue implementado exitosamente en el proyecto. Si bien la FPGA proporciona las ventajas mencionadas, un nivel adicional de optimización podría alcanzarse mediante el diseño de un ASIC (*Application-Specific Integrated Circuit*), que permitiría reducir aún más el tamaño del módulo sincronizador y su consumo de potencia, aspectos cruciales para un sistema portátil. El presente trabajo presenta únicamente el análisis preliminar del ASIC utilizando herramientas de software libre, con el fin de ilustrar el potencial beneficio de esta mejora, aunque no constituye el objetivo principal en la etapa actual del prototipo.

Palabras clave. *UART, MEMS, ASIC, FPGA, Tang Nano 9K.*

ABSTRACT. This work addresses the importance of precise digital signal control for the synchronization between a MEMS (Microelectromechanical Systems) device and a spectrometer within a laboratory prototype. The synchronization was implemented using a Field-Programmable Gate Array (FPGA), whose reconfigurability, high precision, small size, and relatively low cost offer significant advantages over alternatives such as microcontrollers or data acquisition systems (DAQ). The prototype configuration is managed from a computer, and communication with the FPGA is carried out through the UART (Universal Asynchronous Receiver/Transmitter) protocol via a USB connection. Since FPGAs do not include built-in libraries, the UART module had to be designed from scratch, and it was successfully implemented in this project. Although the FPGA provides the aforementioned advantages, an additional level of optimization could be achieved through the design of an Application-Specific Integrated Circuit (ASIC), which would further reduce the size of the synchronizing module and its power consumption, critical aspects for a portable system of this scale. This work presents only a preliminary ASIC analysis using open-source software tools, aiming to illustrate the potential benefits of such an improvement, although it is not the main objective at the current stage of the prototype.

Keywords. *UART, MEMS, ASIC, FPGA, Tang Nano 9K.*

Citación: D. Bouche, E. Acevedo, M. Zambrano, F. Arias, E. Galagarza, "Diseño y validación de ASIC para comunicación UART y control de dispositivos MEMS", *Revista de I+D Tecnológico*, vol. 22, no. 1, pp. (0), 2026.
Tipo de artículo: Original. **Recibido:** 6 de enero de 2026. **Recibido con correcciones:** 23 de marzo de 2026. **Aceptado:** 23 de marzo de 2026.
DOI.

1. Introducción

En el laboratorio de investigación GITTS se ha desarrollado un prototipo basado en la técnica de "un solo píxel" para la adquisición de datos hiperespectrales. En términos generales, el sistema capta la luz reflejada por un objeto o una escena y emplea métodos computacionales para su detección y posterior reconstrucción de la imagen [1].

Para su uso en campo, es necesario que el sistema sea completamente portátil, lo cual implica la miniaturización de sus componentes. Entre los elementos principales del sistema se encuentran dispositivos microelectromecánicos (MEMS) y un espectrómetro.

El término MEMS hace referencia a dispositivos de precisión que combinan componentes mecánicos y eléctricos a escala micrométrica [2]. Estos dispositivos son fabricados principalmente con el uso de materiales semiconductores, incorporando múltiples componentes en un sustrato de silicio [3]. Algunas de las ventajas de los dispositivos MEMS son su tamaño y peso reducidos, bajo consumo energético, facilidad de integración y fabricación en masa [4]. Gracias a estas características se han consolidado como una de las principales soluciones para aplicaciones que requieren sistemas miniaturizados y de alta precisión [2], [5], [6].

Existen diversos sistemas considerados dispositivos MEMS como los DMD (*digital micro-mirrors devices*), que consisten en un arreglo de microespejos y es el utilizado en este proyecto. En la última década, estos sistemas DMD se han ido incorporando a diversas aplicaciones desde el ámbito de las ciencias espaciales [7], [8], [9] hasta experimentos en mecánica cuántica [10].

A través de un sistema óptico, la luz reflejada por la escena es enfocada sobre el arreglo del DMD. Los microespejos del arreglo pueden ser controlados individualmente mediante programación, lo cual facilita generar distintos patrones espaciales. La luz reflejada por estos patrones se redirige hacia el espectrómetro. Cada información por patrón recibida por el espectrómetro es guardada y luego utilizada para reconstruir la imagen de forma indirecta usando lo mismos patrones. Esta técnica se conoce como obtención de imágenes de un solo píxel [1].

El uso de técnicas que requieren alta precisión temporal demanda una sincronización estricta entre los distintos componentes del sistema, en este caso, entre la generación de patrones en el DMD y la adquisición de la señal por parte del espectrómetro. Por ello, resulta necesario emplear plataformas capaces de garantizar un comportamiento determinístico y procesamiento en tiempo real.

Bajo este contexto, la microelectrónica desempeña un papel clave mediante el uso de dispositivos como las FPGA (*Field-Programmable Gate Arrays*). A diferencia de otras plataformas, las FPGA no incorporan librerías predefinidas para aplicaciones específicas, salvo ciertos recursos que se describirán más adelante. En consecuencia, el diseño debe ser desarrollado desde cero por el usuario, configurando el dispositivo a nivel de hardware. En el caso de las FPGA, este diseño se realiza mediante lenguajes de descripción de hardware (HDL), como Verilog. Gracias a esta característica y su reconfigurabilidad, las FPGA son ampliamente utilizadas como plataformas de prototipado y validación de sistemas digitales antes de su implementación como circuitos integrados [11].

La comunicación entre la FPGA y la computadora (PC), desde donde se controlan las operaciones del prototipo, se realiza a través de una conexión USB utilizando el protocolo UART (*Universal Asynchronous Receiver-Transmitter*). Este protocolo permite la transmisión bidireccional de datos sin requerir una señal de reloj compartida, ya que ambos dispositivos establecen previamente una velocidad de comunicación (baudios) para garantizar la correcta sincronización en el envío y la recepción de la información [11], [12], [13], [14], [15], [16], [17].

En comparación con otros protocolos de comunicación comunes, como SPI o I²C, el protocolo UART ofrece una velocidad de transmisión menor, pero logra cubrir distancias mayores [17]. Por esta razón, la longitud del cable USB no afecta de manera significativa la comunicación ni necesita adaptadores adicionales.

Una alternativa a la implementación en FPGA es el diseño e integración de un ASIC (*Application-Specific Integrated Circuit*). La incorporación de un ASIC en el sistema puede ofrecer ventajas adicionales, tales como la optimización del área, mayores velocidades de operación, menor latencia y una reducción significativa del consumo de energía. No obstante, esta opción implica mayores costos de desarrollo y fabricación [18], [19]. En términos generales, el desarrollo de un ASIC suele representar la etapa posterior a la validación completa de un diseño en FPGA, con la diferencia de que el ASIC no es reprogramable.

El presente trabajo se resume en la Figura 1, donde se muestra una representación simplificada de la FPGA en comunicación con la computadora (PC) y los componentes físicos del sistema. La comunicación con la

PC se realiza mediante un módulo que implementa el protocolo UART. Por otro lado, la interacción con los dispositivos físicos se lleva a cabo a través de señales (*triggers*) generadas por un módulo de salida. Este módulo se encarga de activar las señales de control hacia el dispositivo DMD (el cual es un dispositivo basado en MEMS), en sincronía con el espectrómetro.

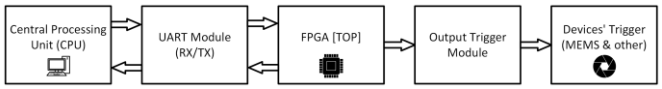

Figura 1. Diagrama de bloques y el flujo de señales entre el host, la interfaz UART de la FPGA y las salidas TTL hacia los dispositivos.

Además, se presentan resultados preliminares del diseño de la FPGA tras su adaptación a una implementación ASIC, con el fin de comparar métricas y parámetros de desempeño entre ambas plataformas.

2. Materiales y Métodos

El diseño del sistema de control de sincronización cuenta con dos partes: el de protocolo de comunicación UART y el de las conexiones de pulsos de salida. Estos están conformados por uno o más módulo dentro de la estructura de HDL.

2.1. Arquitectura del Diseño de Protocolo UART

La comunicación mediante UART es posible gracias a la conexión USB con la placa FPGA. Este protocolo utiliza dos puertos: uno de transmisión (Tx) y otro de recepción (Rx). Estas conexiones ocurren de forma cruzada, es decir el puerto Tx de un dispositivo se conecta con el puerto Rx del otro, y viceversa, como se puede ver en la Figura 2.

En este sistema, la información se envía de manera serial en unidades de 1 *byte* (8 *bits*). A diferencia de otros protocolos, UART no comparte una señal de reloj entre dispositivos; en su lugar, ambas partes deben acordar una tasa de baudios común. Cada dispositivo entonces es responsable de calcular los tiempos necesarios para asegurar la lectura correcta de los datos [20], [21], [22].

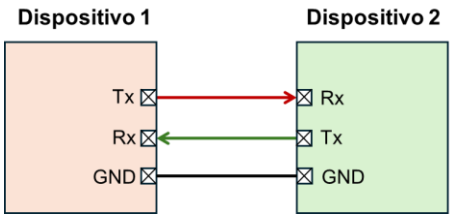

Figura 2. Esquema de conexión UART entre dos dispositivos

La Figura 3 muestra la secuencia de transmisión de un byte mediante el protocolo UART. La transmisión se realiza desde D0 al D7, es decir desde el bit menos significativo (LSB) hasta el bit más significativo (MSB), respectivamente. En condiciones de reposo, la línea de transmisión (Tx) se mantiene en nivel lógico alto. El inicio de la transmisión se indica mediante una transición a nivel lógico bajo, correspondiente al bit de inicio (*Start bit*). A continuación, se envían los bits de datos del byte en el orden descrito. Finalmente, la secuencia concluye con uno o más bits de parada (*Stop bit*) en nivel lógico alto, retornando la línea a su estado de reposo y dejando el sistema preparado para una nueva transmisión [20], [21], [22].

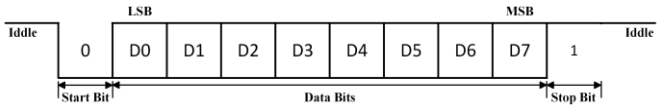

Figura 3. Formato 8N1 (sin bit de paridad) de *trama* UART en transmisión serial desde el LSB al MSB.

Con la Ecuación 1 se determina el número de ciclos de reloj por cada periodo de bit, necesario para sincronizar la transmisión y recepción de datos entre los dispositivos.

$$Pulsos\ de\ baudios\ = \frac{Frequencia\ de\ reloj}{tasa\ de\ baudios} \quad (1)$$

En este proyecto, dicho cálculo se implementa únicamente en la FPGA, ya que en la computadora la comunicación puede gestionarse mediante software que soporte el protocolo UART o a través de bibliotecas integradas, como es el caso de Python.

Durante la ejecución de un experimento, el sistema puede requerir el envío de miles de patrones sobre el objeto para generar la nueva imagen. Cada patrón se mantiene visible durante un tiempo fijo y preestablecido. Es precisamente en esta ventana de actualización cuando la computadora envía el comando correspondiente a la FPGA mediante el protocolo UART para activar las señales necesarias.

Aunque el comando enviado es siempre el mismo, la FPGA lo verifica en cada ciclo y, adicionalmente, envía una respuesta de vuelta a la computadora como mecanismo automático de validación. Este proceso convierte al sistema en una plataforma más robusta, ya que garantiza que la sincronización se mantenga en el momento exacto requerido. Si alguna verificación falla, el sistema suspende inmediatamente la adquisición, pues

cualquier desfase entre los patrones proyectados y los datos capturados invalidaría por completo la muestra. La reconstrucción posterior depende estrictamente de que cada patrón corresponda a la medición correcta.

De este modo, la FPGA no solo recibe información, sino que establece un enlace de comunicación bidireccional con la computadora para garantizar la integridad del proceso. Además, cabe destacar que la ejecución de instrucciones en la FPGA es determinística, gracias a su alta precisión temporal y baja latencia. Por ello, cuando se requiere introducir retardos en el sistema, resulta más confiable implementarlos a nivel de hardware en la FPGA que mediante software en la computadora [23].

2.2. Arquitectura del Diseño de Señales de Salidas para la Sincronización

Esta parte del diseño es la encargada de enviar los pulsos físicos a las conexiones de dipositivo MEMS y el espectrómetro y usar algunos indicadores LEDs de la FPGA para verificación visual de la operación.

Cada vez que la FPGA recibe un comando válido a través del protocolo UART, genera las señales de control necesarias para activar el DMD y el espectrómetro. En el caso del DMD, la señal de disparo corresponde a un pulso en nivel lógico bajo, mientras que el espectrómetro requiere un pulso en nivel lógico alto. Ambos pulsos se generan de forma simultánea y deben presentar una duración suficiente para ser detectados correctamente, manteniendo al mismo tiempo una activación breve que permita el inicio de la operación interna de los dispositivos. Experimentalmente, se ha validado que una duración de 10 ms (milisegundos) resulta adecuada para este propósito.

Para calcular estos tiempos internos en la FPGA, se utiliza como referencia la frecuencia de su reloj interno. Con ello, los retardos requeridos en milisegundos se traducen en un número específico de ciclos de reloj, implementando así temporizaciones precisas y determinísticas.

2.3. Herramientas utilizadas

2.3.1. Tarjeta FPGA

La FPGA utilizada para la configuración del diseño, es la Tang Nano 9K, que posee 27 MHz de frecuencia de reloj interno configurándose a través de Gowin EDA, su software propietario [24], [25]. Para la simulación del diseño se utilizó el software Vivado [26]. Ambos softwares ofrecen versiones académicas las cuales son libre de costo en las cuales se verifica el diseño utilizando HDL.

Finalmente pare el intercambio de datos y validación del envío entre la computadora y la FPGA, se realiza a través de *scripts* de Python desde el software libre Visual Studio Code.

2.3.2. Configuración del ASIC

Para el desarrollo del ASIC se empleó *LibreLane* (previamente conocido como *OpenLane*), una herramienta de código abierto respaldada por Google que permite implementar un diseño descrito en HDL hasta su versión final como circuito integrado [27], [28], [29] . Como se mencionó previamente, este paso adicional se realizó principalmente para demostrar las diferencias en volumen, eficiencia y consumo de energía entre una plataforma de hardware reconfigurable y un circuito integrado de aplicación específica. No obstante, el flujo completo de diseño ASIC implica un proceso más extenso, riguroso y costoso para la etapa actual del proyecto [30].

Además, desde una perspectiva de diseño completo, el módulo UART formaría parte de un sistema integrado más amplio en lugar de implementarse como un chip independiente. Sin embargo, este desarrollo logra validar el funcionamiento del módulo de forma aislada incluso en etapas avanzadas del flujo ASIC, aportando confianza en su integración futura dentro de un sistema mayor.

3. Implementación

El diseño se implementó utilizando Verilog como lenguaje de descripción de hardware (HDL). Al trabajar con HDL, es una buena práctica dividir el sistema en módulos para organizar mejor las funciones y sus responsabilidades.

La Figura 4 muestra el diagrama de bloque del diseño y algunas de las interconexiones entre el módulo principal (*top*) y los submódulos. El módulo *top* integra los componentes encargados del protocolo UART, así como el submódulo de control de las señales conectadas al espectrómetro y al dispositivo MEMS.

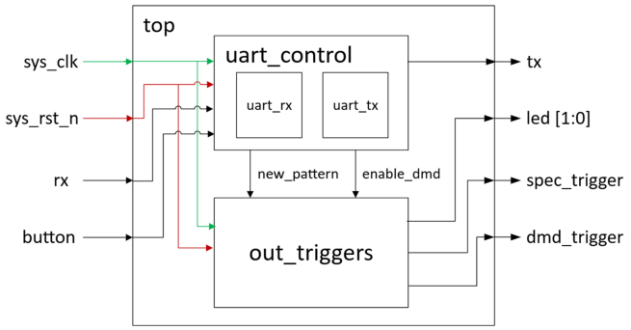

Figura 4. Diagrama de bloque de alto nivel del diseño HDL.

Es común asociar las variables en minúscula en HDL. Las conexiones tx y rx corresponden en este caso a las asociadas a la USB para el protocolo UART. El sistema incluye también un botón (button) para suspender el muestreo durante el experimento. El módulo 'uart _control' maneja la comunicación con la PC y el de 'out _triggers' las conexiones físicas con el espectrómetro y el DMD. Ambos módulos se mantienen comunicadas principalmente cuando un nuevo patrón es registrado.

Para los submódulos de transmisión (Tx) y recepción (Rx) UART, es fundamental considerar los tiempos asociados al protocolo. A partir de la Ecuación 1, se determina el número de ciclos de reloj necesarios para representar la duración de cada bit. En este diseño, se emplea el reloj interno de la FPGA de 27 MHz y una tasa de 115200 baudios, una de las más comunes en comunicaciones con computadoras. Bajo estas condiciones, se obtiene un valor de 234 ciclos de reloj por bit (considerando valores enteros).

Este parámetro resulta especialmente relevante durante la etapa de simulación para verificar que el muestreo de los datos se realice en el instante adecuado. Asimismo, al modificar la frecuencia del reloj o la tasa de baudios, este valor cambia y por ende, se debe ajustar el sistema a la nueva temporización.

A partir de este valor, es posible estimar el tiempo total de transmisión de un dato. En el protocolo UART, cada trama está compuesta por un bit de inicio (*Start bit*), 8 bits de datos y al menos un bit de parada (*Stop bit*), sumando un total de 10 bits. Por lo tanto, el tiempo total de transmisión puede expresarse como en la Ecuación 2:

$$Tiempo = \frac{Pulsos\ de\ baudios * 10}{Frecuencia\ de\ reloj} \qquad (2)$$

Para este caso, si se reemplaza los Pulsos de baudios por 234 y la Frecuencia de reloj por 27 MHz, el resultado es aproximadamente 86.7 µs (microsegundos). Este valor coincide (o se aproxima) al obtenido al invertir la tasa de baudios y multiplicarla por el número total de bits transmitidos. Este análisis facilita dimensionar los tiempos de operación del sistema y garantizar la correcta sincronización entre los dispositivos.

Cabe destacar que el diseño se concibió con un enfoque flexible, de modo que la frecuencia de reloj y la tasa de baudios pueden modificarse desde el módulo principal. De este modo el sistema se adapta automáticamente a diferentes configuraciones, ajustando su velocidad de operación según los requerimientos de la aplicación.

Por otra parte, el submódulo de recepción (Rx) requiere especial atención, ya que debe realizar una lectura precisa de cada bit recibido. Dado que la señal Rx es asíncrona respecto al reloj de la FPGA, se implementa una etapa de sincronización para alinearla con el dominio de reloj interno y evitar errores de validación durante las transiciones de la señal. Posteriormente, se emplea un mecanismo de detección del bit de inicio (*Start bit*) en la señal Rx sincronizada. Una vez detectado este evento, se introduce un retardo equivalente a la mitad del periodo de bit, con el fin de posicionar el muestreo en el centro del primer bit de datos.

A partir de este punto, los bits siguientes se muestrean en intervalos iguales al periodo de bit previamente calculado. Así, cada lectura se realiza en la región más estable de la señal, reduciendo la probabilidad de errores debidos a variaciones temporales o ruido en la línea de comunicación.

La Figura 5 muestra la etapa final de la recepción del carácter 'D', cuyo valor en código ASCII [31] fue seleccionado como comando válido para activar la sincronización de las señales dirigidas al MEMS y al espectrómetro. En representación binaria, este carácter corresponde a 01000100. La Figura 5(c) ilustra la secuencia de recepción de los bits, enumerados hasta el noveno bit. Adicionalmente, se observa el proceso de reconstrucción del byte en la Figura 5(d), donde los bits se almacenan progresivamente desde el menos significativo al más significativo. Una vez recibido el octavo bit de datos, la información queda completamente registrada, mientras que el noveno bit corresponde al bit de parada (*Stop bit*), indicando el final de la transmisión.

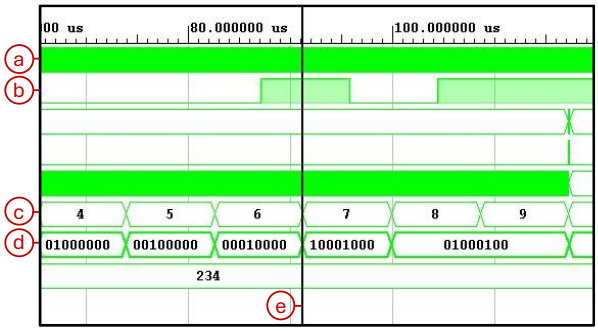

Figura 5. Simulación de la recepción del carácter 'D' en código ASCII. (a) Señal de reloj (clock); (b) señal de recepción (Rx); (c) contador de bits recibidos; (d) registro de reconstrucción de datos en formato binario; (e) señal de muestreo indicando la lectura en la mitad del periodo de bit.

La frecuencia del reloj no es visible en esta vista de la simulación, ya que el nivel de vista se centró específicamente en el periodo asociado a la tasa de baudios. Más adelante se presenta un acercamiento con mayor detalle en el cual sí es posible observar claramente la señal del reloj.

En la Figura 6 se observa la FPGA configurada dentro del prototipo del laboratorio para llevar a cabo las pruebas.

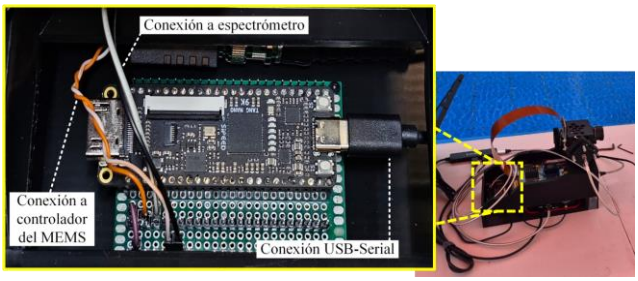

Figura 6. Conexiones de la tarjeta FPGA dentro del prototipo.

Para la configuración del ASIC del diseño usando LibreLane, es necesario contar con un entorno Linux, como Ubuntu. En Windows, puede utilizarse mediante una máquina virtual o WSL2 sin reemplazar el sistema operativo principal. La instalación se realiza siguiendo las instrucciones del repositorio oficial en GitHub, que incluyen clonar el proyecto [28]. Una vez instalado, los archivos Verilog del diseño pueden cargarse en la herramienta. Tras completar el flujo, OpenLane genera reportes intermedios y produce como salida un archivo GDSII listo para fabricación.

4. Resultados y discusión

Al hacer un acercamiento de la Figura 5, se obtiene la Figura 7, donde los pulsos de reloj se distinguen con mayor claridad. Para facilitar la interpretación, se presenta el carácter ASCII recibido ('D') en lugar de su representación binaria. En la figura, se observa el conteo de ciclos de reloj (resaltado en rojo), el cual se reinicia al alcanzar el valor de 234. Asimismo, en color negro se muestra que, al finalizar esta secuencia, el dato recibido se mantiene válido junto con una señal de completado activa durante un ciclo de reloj. Esta señal, en conjunto con el dato recibido, es utilizada para generar los pulsos de disparo dirigidos al dispositivo MEMS y al espectrómetro.

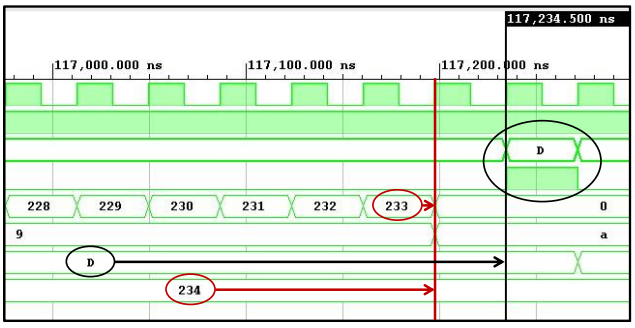

Figura 7. Simulación de la recepción del carácter 'D' en ASCII acercada.

En la simulación, se incorporó un mecanismo de respuesta automática para detectar la recepción de un carácter válido, en este caso 'D'. Como se muestra en la Figura 8, una vez identificado este carácter, se genera una señal de confirmación que es transmitida de regreso, verificando así el correcto funcionamiento del enlace de comunicación. El tiempo observado en la figura corresponde únicamente al entorno de simulación y se utiliza como referencia para validar la secuencia de eventos.

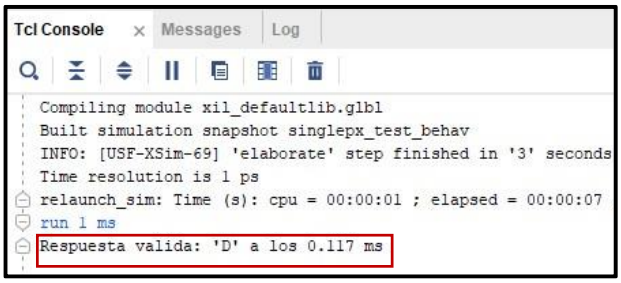

Figura 8. Respuesta automática de la simulación cuando un carácter válido es recibido

Para el diseño del submódulo de transmisión (Tx) UART es el mismo principio pero sin hacer el desfase al inicio. Se envían los bits del menos significativo al más significativo incluyendo el bit de inicio y del final. El dispositivo receptor se encarga de reconstruir esta

secuencia para obtener la información completa, tal como se describió previamente para el submódulo de recepción (Rx).

Aunque el diseño opera inicialmente con una frecuencia de reloj de 27 MHz y una tasa de 115200 baudios, también fue evaluado a frecuencias y velocidades superiores. La FPGA dispone de recursos internos como el PLL (Phase-Locked Loop), permitiendo generar nuevas frecuencias de reloj a partir de la señal base. En este caso, se configuró el sistema para triplicar la frecuencia de operación hasta 81 MHz. Habilitando así probar con tasas de transmisión mayores como la de 921600 baudios, la cual es un valor común en interfaces UART de alto rendimiento.

Los resultados de esta prueba demuestran la robustez del diseño ante mayores exigencias de comunicación y temporización, como se observa en la Figura 9. Aplicando la Ecuación 1, se obtiene un valor de 87 ciclos de reloj por bit (considerando valores enteros) para estas nuevas condiciones.

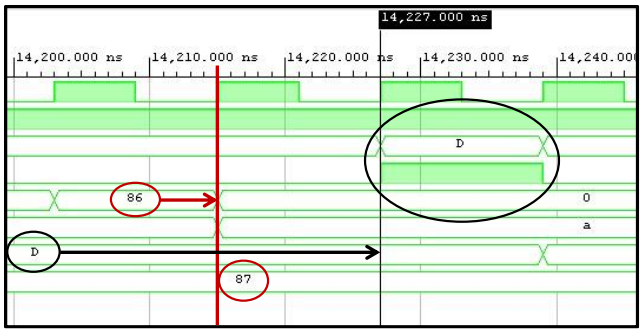

Figura 9. Simulación de sobre la recepción del carácter 'D' en ASCII acercada, a mayor frecuencia y tasa de baudios

Al igual que en la prueba anterior, el contador de ciclos de reloj se reinicia al alcanzar el valor de 87 (resaltado en rojo), habilitando la recepción del byte junto con un pulso de completado (en negro). Para estimar la duración de esta transmisión junto a la nueva frecuencia, se aplica nuevamente la Ecuación 2, obteniendo un aproximado de 10.7 µs. Este resultado representa una reducción significativa respecto al caso anterior (86.7 µs), evidenciando la mejora en la velocidad de transmisión al incrementar la frecuencia de operación y la tasa de baudios.

Similarmente, la Figura 10 muestra la impresión del mensaje válido para este caso, el cual se obtiene en un menor tiempo dentro del mismo entorno de simulación. Este resultado confirma el correcto funcionamiento del sistema bajo condiciones de operación más exigentes. No obstante, debido a que el incremento en la tasa de transmisión no es un requisito crítico para esta aplicación, se mantienen los valores previamente definidos con el fin de preservar la eficiencia energética [32].

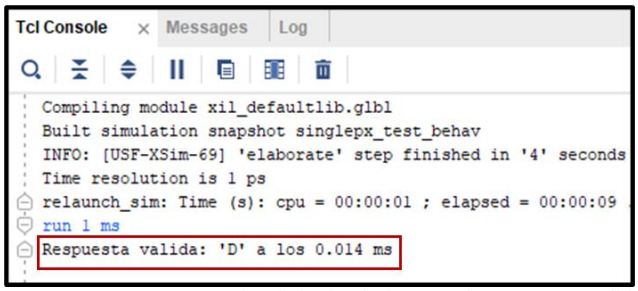

Figura 10. Respuesta automática de la simulación cuando un carácter válido es recibido a mayor velocidad

4.1. Resultados de la Configuración en la FPGA

Para la implementación en FPGA, la Tabla 1 resume los recursos utilizados y los reportes generados por el software de síntesis.

Tabla 1. Reporte de utilización y consumo del diseño en la FPGA

Propiedad	**Valor/Cantidad**
Latches generados	Ninguno
LUTs utilizados	217
Registros utilizados:	
- *Flip-flops* lógicos	108
- *Flip-flops* de entrada/salida	5
- Total de *flip-flops*	113
Consumo de energía (mW):	
- Chip principal	26.7
- PSRAM de la tarjeta	86
- Consumo total	112.7

El diseño presenta un uso eficiente de recursos lógicos, destacando la ausencia de *latches*, lo cual indica que no existen elementos de almacenamiento no controlados por reloj, evitando comportamientos indeterminados. Las LUTs (*Look-Up Tables*) corresponden a los bloques combinacionales básicos de la FPGA, y su bajo número refleja una implementación compacta del diseño. Por su parte, los flip-flops representan los elementos secuenciales utilizados para almacenamiento sincronizado, siendo fundamentales para garantizar un comportamiento temporal determinístico.

En conjunto, estos resultados evidencian un bajo uso de recursos y un consumo energético reducido, haciendo del diseño un sistema adecuado para aplicaciones

embebidas y escenarios con requerimientos de sincronización precisa.

La Figura 11 muestra el prototipo físico del sistema usando FPGA para la sincronización. Dicho sistema fue validado experimentalmente, demostrando una correcta sincronización entre el dispositivo MEMS y el espectrómetro. El comportamiento observado permitió la ejecución consistente del sistema en condiciones reales, incluyendo la aplicación de múltiples patrones por experimento sin errores de temporización.

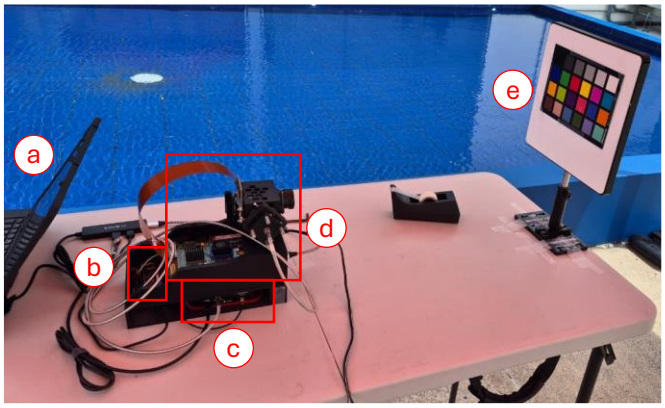

Figura 11. Prototipo de cámara hiperespectral de un solo píxel. (a) Computadora; (b) FPGA; (c) Espectrómetro; (d) Dispositivo MEMS; (e) Área a ser capturada.

4.2. Resultados de la Configuración en ASIC

Luego de validar el diseño por simulación y en la FPGA, se procedió a sintetizarlo como ASIC utilizando LibreLane, una herramienta de código abierto para flujo completo de diseño físico.

Tras completar el flujo de diseño, la herramienta generó el archivo GDSII, formato estándar utilizado para la fabricación de circuitos integrados o "dado" en esta fase. Este resultado se muestra en la Figura 12, donde se observa el enrutamiento automático del diseño de acuerdo con las restricciones de densidad establecidas. Debido a la naturaleza compacta del sistema, no se presentaron problemas de colocación ni congestión durante la etapa de implementación física.

Adicionalmente, la herramienta generó diversos reportes de síntesis física, cuyos parámetros más relevantes se resumen en la Tabla 2. Las celdas utilizadas corresponden principalmente a compuertas lógicas estándar (AND, OR, NAND, NOR, entre otras) y elementos secuenciales como *flip-flops*.

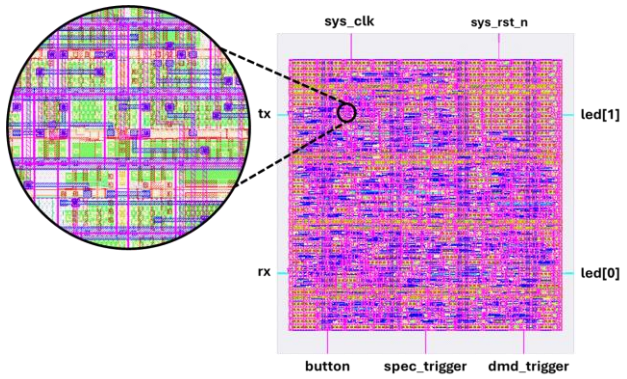

Figura 12. Esquema del prototipo ASIC generado con la herramienta OpenLane (vista del GDSII generado).

Tabla 2. Reporte de síntesis física del diseño para ASIC

Propiedad	**Valor/Cantidad**
Infracciones en el diseño	Ninguno
Celdas generadas	1974
Flip-flops totales	16
Área total del dado (mm^2)	0.02
Consumo de energía (mW):	
- Consumo interno típico	0.369
- Conmutación típico	0.209
- Consumo total	0.578

Los resultados obtenidos evidencian una implementación altamente integrada, sin infracciones de diseño, indicando el cumplimiento de las reglas físicas establecidas. Asimismo, el reducido número de elementos secuenciales y el área total del dado reflejan la simplicidad estructural del sistema. En términos energéticos, el consumo reportado se mantiene en niveles bajos, coherente con una implementación dedicada. En comparación con la FPGA, estas mejoras se deben a la eliminación de la sobrecarga asociada a la reconfigurabilidad, lo que permite una mayor densidad de integración y una reducción significativa en el consumo de energía.

5. Conclusiones

El presente trabajo demuestra la viabilidad de una arquitectura basada en UART para la sincronización determinista de un espectrómetro y dispositivos MEMS, validada mediante simulación y una implementación funcional en FPGA. El diseño se caracteriza por su bajo costo computacional, estabilidad temporal y capacidad de adaptación a distintos parámetros de operación, lo que facilita su integración en configuraciones experimentales y prototipos de laboratorio.

La FPGA se consolidó como una plataforma adecuada para este tipo de aplicaciones, al ofrecer flexibilidad para iterar sobre el diseño, depurar el protocolo y realizar pruebas en tiempo real. De manera complementaria, el análisis mediante un flujo abierto de diseño ASIC permitió evidenciar que el sistema posee un nivel de integración y eficiencia suficiente para su posible implementación en entornos con restricciones de área y consumo, aunque este no constituye el objetivo principal del trabajo.

Las métricas obtenidas, tanto en consumo energético como en utilización de recursos, proporcionan una referencia práctica para el dimensionamiento de futuras implementaciones, especialmente en aplicaciones donde la temporización precisa y una comunicación confiable son críticas. En su estado actual, el sistema cumple con los requerimientos del proyecto, contribuyendo a la reducción de costos, tamaño y a la portabilidad del prototipo.

Como línea de trabajo futuro, se plantea la expansión del sistema hacia arquitecturas de control más complejas, aprovechando la reutilización de los módulos desarrollados, particularmente los bloques de comunicación UART. Asimismo, en escenarios que demanden una mayor optimización energética o integración, el flujo ASIC presentado ofrece una ruta viable para una eventual migración del diseño

AGRADECIMIENTOS

Al Grupo de investigación en tecnologías avanzadas de telecomunicaciones y procesamiento de señales (GITTS) por bridar el espacio y las herramientas necesarias para la realización de este trabajo.

CONFLICTO DE INTERESES

Los autores declaran no tener algún conflicto de interés.

CONTRIBUCIÓN Y APROBACIÓN DE LOS AUTORES

Diego Bouche: Conceptualización, Metodología, Software, Redacción - Revisión y Edición. Edwin Acevedo: Investigación, Redacción - Preparación del borrador original. Maytee Zambrano: Redacción – Revisión y Edición. Fernando Arias: Redacción – Revisión y Edición. Edson Galagarza: Redacción – Revisión y Edición.

Todos los autores afirmamos que hemos leído y aprobado la versión final de este artículo.

REFERENCIAS

[1] G. M. Gibson, S. D. Johnson, y M. J. Padgett, «Single-pixel imaging 12 years on: a review», *Opt. Express*, vol. 28, n.º 19, p. 28190, sep. 2020, doi: 10.1364/OE.403195.

[2] C. Chircov y A. M. Grumezescu, «Microelectromechanical Systems (MEMS) for Biomedical Applications», *Micromachines (Basel).*, vol. 13, n.º 2, p. 164, ene. 2022, doi: 10.3390/mi13020164.

[3] A. M. Basuwaqi, M. H. M. Khir, A. Y. Ahmed, Almur. A. S. Rabih, M. U. Mian, y J. O. Dennis, «Effects of frequency and voltage on the output of CMOS-MEMS device», en *2017 IEEE Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PrimeAsia)*, IEEE, oct. 2017, pp. 49-52. doi: 10.1109/PRIMEASIA.2017.8280361.

[4] Y. Ziying, G. Hong, y J. Wenhao, «Design and implementation of a MEMS-based attitude angle measuring system for moving objects», en *2021 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS)*, IEEE, jul. 2021, pp. 145-149. doi: 10.1109/ICPICS52425.2021.9524095.

[5] A. S. Algamili *et al.*, «A Review of Actuation and Sensing Mechanisms in MEMS-Based Sensor Devices», *Nanoscale Res. Lett.*, vol. 16, n.º 1, p. 16, ene. 2021, doi: 10.1186/s11671-021-03481-7.

[6] P. Pattanaik y M. Ojha, «Review on challenges in MEMS technology», *Mater. Today Proc.*, vol. 81, pp. 224-226, 2023, doi: 10.1016/j.matpr.2021.03.142.

[7] W. Snyder, G. M. Halferty, D. Vorobiev, D. Chafetz, y B. T. Fleming, «Designing a compact, self-contained control and power system for a DMD-based spectrograph suitable for the space environment», en *Space Telescopes and Instrumentation 2024: Ultraviolet to Gamma Ray*, J.-W. A. den Herder, K. Nakazawa, y S. Nikzad, Eds., SPIE, ago. 2024, p. 155. doi: 10.1117/12.3019136.

[8] F. Yalcinkaya, T. Koc, y Z. Pala, «Spatial light modulator design and generation of structured electromagnetic waves using digital light processors», *Optica Applicata*, vol. 52, n.º 3, 2022, doi: 10.37190/oa220311.

[9] A. Kumar, S. P. K., P. G. Deshmukh, y P. D. V. S., «Digital Micro-Mirror Device (DMD) controller development for INSIST mission», en *Space Telescopes and Instrumentation 2024: Ultraviolet to Gamma Ray*, J.-W. A. den Herder, K. Nakazawa, y S. Nikzad, Eds., SPIE, ago. 2024, p. 188. doi: 10.1117/12.3017737.

[10] M. Mathews, X. Yin, A. Baumgartner, T. Esslinger, y A. Akin, «A Control and Testing Framework for a Digital Micromirror Device in Neutral Atom Quantum Computing and Simulation Platforms», en *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, IEEE, sep. 2022, pp. 724-726. doi: 10.1109/QCE53715.2022.00098.

[11] Z. Du, Y. Liu, C. Qiu, y X. Zhang, «Verilog implementation of configurable UART module», en *Second International*

Conference on Statistics, Applied Mathematics, and Computing Science (CSAMCS 2022), S. Jin y W. Dai, Eds., SPIE, mar. 2023, p. 143. doi: 10.1117/12.2672692.

[12] L. Xu, Y. Qi, Z. Jiang, y X. Wei, «Fast frequency relocking for synchronization enhanced resonant accelerometer», *Microsyst. Nanoeng.*, vol. 8, n.º 1, p. 93, sep. 2022, doi: 10.1038/s41378-022-00428-5.

[13] W. Huang y G. Sheng, «Analysis and Research on UART Communication Protocol», en *2024 4th Asia-Pacific Conference on Communications Technology and Computer Science (ACCTCS)*, IEEE, feb. 2024, pp. 768-771. doi: 10.1109/ACCTCS61748.2024.00140.

[14] P. Sivaranjani, S. Sasikala, A. Lavanya, y M. Keerthana, «Design and Analysis of UART Protocol with Sec-Ded and Implementation on FPGA», en *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, IEEE, jul. 2023, pp. 1-8. doi: 10.1109/ICCCNT56998.2023.10306744.

[15] A. K. Gupta, A. Raman, N. Kumar, y R. Ranjan, «Design and Implementation of High-Speed Universal Asynchronous Receiver and Transmitter (UART)», en *2020 7th International Conference on Signal Processing and Integrated Networks (SPIN)*, IEEE, feb. 2020, pp. 295-300. doi: 10.1109/SPIN48934.2020.9070856.

[16] K. B. Sowmya, S. Gomes, y V. R. Tadiparthi, «Design of UART Module using ASMD Technique», en *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, IEEE, jun. 2020, pp. 176-181. doi: 10.1109/ICCES48766.2020.9138098.

[17] J. Chen y S. Huang, «Analysis and Comparison of UART, SPI and I2C», en *2023 IEEE 2nd International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA)*, IEEE, feb. 2023, pp. 272-276. doi: 10.1109/EEBDA56825.2023.10090677.

[18] A. J M y T. Anujan, «Design and ASIC Implementation of Area Efficient UART Core in SCL 180nm Technology», en *2024 28th International Symposium on VLSI Design and Test (VDAT)*, IEEE, sep. 2024, pp. 1-5. doi: 10.1109/VDAT63601.2024.10705725.

[19] D. Mayer, M. Lehmann, V. Beyer, A. Schneider, y P. Schneider, «Challenges in Design and Validation of MEMS Based Smart Sensor Systems», en *2024 Symposium on Design, Test, Integration and Packaging of MEMS/MOEMS (DTIP)*, IEEE, jun. 2024, pp. 1-6. doi: 10.1109/DTIP62575.2024.10613045.

[20] L. Cao, J. Chen, y J. Li, «Working principle and application analysis of UART», en *2023 IEEE 2nd International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA)*, IEEE, feb. 2023, pp. 255-259. doi: 10.1109/EEBDA56825.2023.10090571.

[21] S. Harutyunyan, T. Kaplanyan, A. Kirakosyan, y A. Momjyan, «Design And Verification Of Autoconfigurable UART Controller», en *2020 IEEE 40th International Conference on Electronics and Nanotechnology (ELNANO)*, IEEE, abr. 2020, pp. 347-350. doi: 10.1109/ELNANO50318.2020.9088789.

[22] E. Peña y M. G. Legaspi, «UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter», 2020.

[23] R. R. Pahlevi, A. G. Putrada S., y M. Abdurohman, «Fast UART and SPI Protocol for Scalable IoT Platform», en *2018 6th International Conference on Information and Communication Technology (ICoICT)*, IEEE, may 2018, pp. 239-244. doi: 10.1109/ICoICT.2018.8528745.

[24] GOWIN Semiconductor Corp., «Gowin FPGA Designer (Version 1.9) [Software]», https://www.gowinsemi.com/en/support/home.

[25] «GW1NR series of FPGA Products». [En línea]. Disponible en: www.alcom.be

[26] AMD Xilinx, «Vivado Design Suite (Version 2024.2) [Software]», https://www.xilinx.com/support/download.html.

[27] M. Fauzan *et al.*, «Physical Design of RISC-V based System-on-Chip using OpenLane», en *2024 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, IEEE, nov. 2024, pp. 529-533. doi: 10.1109/APCCAS62602.2024.10808656.

[28] M. Shalan y T. Edwards, «Building OpenLANE», en *Proceedings of the 39th International Conference on Computer-Aided Design*, New York, NY, USA: ACM, nov. 2020, pp. 1-6. doi: 10.1145/3400302.3415735.

[29] W. Zhi, T. Yi, y Z. Hong, «A Review and Perspective on Electrode Patch-Based Fetal ECG Monitoring ASIC», en *2021 IEEE 14th International Conference on ASIC (ASICON)*, IEEE, oct. 2021, pp. 1-4. doi: 10.1109/ASICON52560.2021.9620269.

[30] B. J. Kang *et al.*, «A survey of FPGA and ASIC designs for transformer inference acceleration and optimization», *Journal of Systems Architecture*, vol. 155, p. 103247, oct. 2024, doi: 10.1016/j.sysarc.2024.103247.

[31] International Organization for Standardization [ISO], «Information technology — ISO 7-bit coded character set for information interchange (ISO/IEC 646:1991)», 1991.

[32] Bishwajeet Pandey, Keshav Kumar, Aiza Batool, y Shabeer Ahmad, «Implementation of power-efficient control unit on ultrascale FPGA for green communication», *3c Tecnología: glosas de innovación aplicadas a la pyme*, vol. 10, n.º 1, pp. 93-105, 2021.