TeCEO
Revista Tecnología, Ciencia y Estudios Organizacionales

TeCEO, 6(12), 2024.
Revista Tecnología, Ciencia y Estudios Organizacionales / ISSN: 2594-1410
https://doi.org/10.56913/teceo.6.12.193-205

Procesamiento Paralelo de 2 FPGAs para Control y Aplicaciones Autotrónicas.

Parallel Processing of 2 FPGAs for Control and Autronic Applications.

- **Óscar Villanueva-Medina,** es alumno de la Universidad Politécnica de Guanajuato (México) (21030559@upgto.edu.mx), (https://orcid.org/0009-0005-9345-7070), Licenciado.
- **Julio César Peña-Aguirre**, es profesor en la Universidad Politécnica de Guanajuato (México) (jcpena@upgto.edu.mx), (https://orcid.org/0000-0002-6211-7760), Doctor.
- **Yosafat Jetsemani Samano-Flores,** es profesor en la Universidad Politécnica de Guanajuato, (México) (ysamano@upgto.edu.mx), (https://orcid.org/0000-0003-4173-6236), Máster.
- **Tomás Serrano-Ramírez,** es profesor en la Universidad Politécnica de Guanajuato (México) (juan.dd@matehuala.tecnm.mx), (https://orcid.org/0000-0001-9116-0242), Máster.

Resumen: El presente estudio analiza la implementación de un sistema de procesamiento paralelo utilizando dos FPGAs en aplicaciones de control autotrónico. Tradicionalmente, los sistemas de control automotriz enfrentan limitaciones debido a la capacidad de una sola FPGA para procesar grandes volúmenes de datos en tiempo real. Para superar estas limitaciones, se ha desarrollado un sistema que emplea dos FPGAs en paralelo, logrando una mejora significativa en la eficiencia y capacidad de procesamiento. La metodología utilizada incluye la programación en VHDL, la comunicación entre FPGAs y el uso de una ESP32 para medir el tiempo de procesamiento. Los resultados indican una reducción del 50% en el tiempo de lectura de datos, lo que representa un avance crucial en la eficiencia de los sistemas de control automotriz avanzados.

Palabras clave: FPGA, procesamiento paralelo, control autotrónico, ESP32, VHDL.
Cómo citar: Villanueva-Medina, O.; Peña-Aguirre, J.C.; Samano-Flores, Y.J.; y Serrano-Ramírez, T. (2024). Procesamiento Paralelo de 2 FPGAs para Control y Aplicaciones Autotrónicas**.** *Tecnología, Ciencia y Estudios Organizacionales*, 6(12), pp. 193-205. https://doi.org/10.56913/teceo.6.12.193-205

Recepción: 14-10-2024

Aprobación: 23-10-2024

Abstract: This study analyzes the implementation of a parallel processing system using two FPGAs in autronic control applications. Traditionally, automotive control systems face limitations due to the processing capacity of a single FPGA to handle large volumes of real-time data. To overcome these limitations, a system was developed employing two FPGAs in parallel, significantly improving processing efficiency and capacity. The methodology includes programming in VHDL, FPGA communication, and the use of an ESP32 to measure processing time. Results indicate a 50% reduction in data reading time, representing a crucial advancement in the efficiency of advanced automotive control systems.

Keywords: FPGA, parallel processing, autronic control, ESP32, VHDL.

Introducción – Procesamiento Paralelo n FPGAs para Aplicaciones Autotrónicas

El rápido avance de la tecnología en la industria automotriz ha generado una creciente demanda por sistemas de control más rápidos, eficientes y precisos. En este contexto, los sistemas autotrónicos, que combinan la automatización y la electrónica, requieren soluciones de procesamiento capaces de manejar grandes volúmenes de datos en tiempo real. Las Field-Programmable Gate Arrays (FPGAs) han demostrado ser una herramienta valiosa para satisfacer estas demandas debido a su capacidad de ser reconfiguradas para ejecutar tareas específicas con alta eficiencia (Guo et al., 2016). Sin embargo, la creciente complejidad de los sistemas automotrices plantea nuevos retos, y las limitaciones de procesamiento de una sola FPGA comienzan a ser un obstáculo para tareas críticas como el control en tiempo real y la toma de decisiones en vehículos autónomos.

Diversos estudios han explorado el uso de FPGAs en aplicaciones autotrónicas debido a su capacidad para ofrecer un procesamiento eficiente y en tiempo real con un bajo consumo energético. En (Cabanes et al., 2019) propusieron una metodología de diseño y prototipado basada en Multi-CPU/FPGA para la detección y reconocimiento de objetos en vehículos autónomos, logrando una aceleración de hasta 300 veces en comparación con soluciones puramente basadas en software. De manera similar, (Hao et al., 2019) desarrollaron un sistema híbrido GPU+FPGA que permite a la FPGA detectar fallos del sistema primario y ejecutar tareas de respaldo, mejorando así la confiabilidad y autonomía del vehículo.

En (Kasem et al., 2021) presentaron un estudio que revisa las soluciones basadas en inteligencia artificial implementadas en FPGAs para vehículos autónomos, destacando su eficiencia en comparación con GPUs y CPUs. En la misma línea, (Kojima & Nose, 2018) desarrollaron un vehículo robótico autónomo utilizando una FPGA Zynq 7020, con procesamiento de imagen y control de motores implementados en lógica programable y en el sistema de procesador. Por su parte, (Li et al., 2022) propusieron una solución basada en FPGA para la implementación de control predictivo, lo que facilita su uso en investigación gracias a una reducción significativa en la complejidad de desarrollo.

En (Wei et al., 2018) implementaron redes neuronales binarizadas (BNNs) en FPGAs para el reconocimiento de peatones y obstáculos, mejorando tanto la velocidad como la precisión en la detección. Cesarano (Cesarano, 2018) investigó la implementación del acelerador NVIDIA Deep Learning Accelerator (NVDLA) en FPGAs para la inferencia de aprendizaje profundo, mostrando cómo la modularidad y configurabilidad del sistema puede optimizar su uso en vehículos autónomos. En (Hasegawa et al., 2019; Humaidi et al., 2018) propusieron una arquitectura de detección de carriles basada en FPGA, optimizando el procesamiento de imágenes en tiempo real para aumentar la seguridad de los vehículos autónomos.

Finalmente, Ghielmetti (Ghielmetti et al., 2022) investigaron cómo las FPGAs pueden servir como aceleradores de hardware para la segmentación semántica en tiempo real, logrando una latencia de 4.9 ms por imagen en una implementación completamente integrada en chip. En conjunto, estos estudios destacan el papel crucial de las FPGAs como una plataforma eficiente y flexible para enfrentar los desafíos de los sistemas autónomos, superando las limitaciones de otros aceleradores en términos de consumo de energía y latencia.

El uso de FPGAs en sistemas de control industrial y autotrónico ha ganado terreno gracias a su capacidad para procesar tareas en paralelo y su flexibilidad en aplicaciones de tiempo real. La arquitectura paralela de las FPGAs permite ejecutar múltiples operaciones simultáneamente,

optimizando el rendimiento y la velocidad en sistemas que requieren alta precisión, como los de control en tiempo real. Shen y Xiao (M. Shen & Xiao, 2020) muestran cómo las FPGAs son especialmente útiles en controladores de motores y sensores, donde las demandas de procesamiento en tiempo real son críticas. Investigaciones en el ámbito de control de motores han demostrado que las FPGAs pueden mejorar significativamente el rendimiento al gestionar eficientemente señales de control y feedback de manera paralela (Rodríguez et al., 2020; M. Shen et al., 2021; M. Shen & Xiao, 2020).

En particular, los avances recientes en la integración de inteligencia embebida en FPGAs han ampliado sus aplicaciones en sistemas autónomos y de computación en el borde, mejorando la eficiencia energética y la latencia gracias a su capacidad de personalización (Seng et al., 2021). Por ejemplo, el uso de arquitecturas reconfigurables permite la implementación de redes neuronales y algoritmos de aprendizaje profundo en FPGAs, superando las limitaciones tradicionales de latencia y consumo energético en comparación con CPUs y GPUs (Adil Yazdeen et al., 2021; Zhang et al., 2022).

Además, la arquitectura paralela de las FPGAs también se ha implementado con éxito en algoritmos avanzados como los algoritmos genéticos, que requieren un alto nivel de procesamiento simultáneo. En este contexto, un estudio realizado por Zhang (Zhang et al., 2022). y Cao (Cao et al., 2022) muestra cómo los algoritmos paralelos en FPGAs pueden reducir considerablemente los tiempos de procesamiento y mejorar la eficiencia, lo que es particularmente relevante para aplicaciones que necesitan gestionar grandes volúmenes de datos en tiempo real. Adicionalmente, en el trabajo de (Guo et al., 2016) demuestran que el procesamiento paralelo en FPGAs no solo es viable, sino que también puede superar las limitaciones de una sola FPGA, optimizando la sincronización y el manejo de datos complejos en aplicaciones críticas como los sistemas de control automotriz.

Este estudio tiene como objetivo implementar y evaluar un sistema de procesamiento paralelo utilizando dos FPGAs, con el apoyo de una ESP32 para medir los tiempos de procesamiento y optimizar los resultados. La ESP32 juega un papel crucial al permitir una medición precisa de los tiempos de ejecución, lo que facilita la evaluación del rendimiento en tiempo real de las FPGAs. En literatura se reporta que, la tarjeta ESP32 se destacó como una herramienta clave en la implementación de sistemas de monitoreo en conjunto con la FPGA. En el trabajo (Nur-A-Alam et al., 2021), se utilizó la ESP32 para establecer comunicación inalámbrica a través de LoRa en un sistema de monitoreo y control de electrodomésticos a larga distancia. Este sistema IoT demostró ser eficiente en términos de consumo energético y precisión, alcanzando hasta 12 km de distancia de operación. Por otro lado, en (Bosse, 2022), se empleó la ESP32 en redes distribuidas de sensores y sistemas de cómputo paralelos. El marco propuesto integra nodos virtuales y físicos, facilitando el monitoreo y control de estos sistemas complejos a través de máquinas virtuales y un entorno web accesible. Ambos trabajos ejemplifican cómo la ESP32, en combinación con FPGA y tecnologías de virtualización, permite la creación de soluciones avanzadas para la automatización y el monitoreo distribuido.

La investigación se centra en demostrar que la distribución de tareas entre dos FPGAs puede reducir los tiempos de procesamiento en un 50%, mejorando así la eficiencia de los sistemas autotrónicos. Como se reporta en (Dias et al., 2020), quienes implementaron el algoritmo K-Means en una FPGA de manera paralela, logrando una aceleración significativa en la clasificación de datos al dividir las tareas entre múltiples unidades de procesamiento. De manera similar, (Kastner et al., 2018) exploraron técnicas de programación paralela en FPGAs, destacando la capacidad de

estas arquitecturas para ejecutar múltiples operaciones simultáneamente y así mejorar la eficiencia en tareas computacionalmente intensivas. Por su parte, (Mueller et al., 2009) investigaron el procesamiento de datos en FPGAs, donde el paralelismo permitió manejar grandes conjuntos de datos de manera eficiente, mientras que (Stojilović, 2017) realizó un estudio sobre el enrutamiento paralelo en FPGAs, abordando los retos y avances en la mejora del rendimiento en el diseño de hardware. Estos trabajos demuestran cómo el procesamiento paralelo en FPGAs ha sido fundamental para acelerar procesos complejos en diversas aplicaciones.

El enfoque de este trabajo busca no solo validar el potencial del procesamiento paralelo en FPGAs, sino también superar los retos técnicos asociados a la sincronización y comunicación de los dispositivos, utilizando herramientas como el protocolo CAN-Bus para conectar las FPGAs con la ESP32.

El protocolo CAN-BUS (Controller Area Network) es un estándar de comunicación utilizado ampliamente en sistemas embebidos para la transmisión de datos entre múltiples nodos en tiempo real, con aplicaciones en automóviles, sistemas industriales y dispositivos de control (Nesbø et al., 2020; Pawar, 2022). Este protocolo se caracteriza por su alta fiabilidad y eficiencia en la transmisión de mensajes, lo que lo convierte en una opción ideal para sistemas críticos. En el estudio (Hronek, 2023), se destaca el uso de CAN-BUS para pruebas automáticas de latencia, subrayando su importancia en entornos de prueba y evaluación continua. En (Huang et al., 2024) examinan cómo el protocolo se utiliza en la actualización remota de FPGAs, demostrando su flexibilidad en la gestión de sistemas complejos. Asimismo, en los trabajos de (Pawar, 2022; Shimiao et al., 2020) exploran su implementación en FPGAs y microsatélites, resaltando la seguridad y confiabilidad del protocolo en aplicaciones espaciales y de control avanzado

Los resultados de este estudio aportarán nuevas perspectivas para el desarrollo de soluciones más eficientes en sistemas de control automotriz avanzados y contribuirán a la investigación en hardware reconfigurable para aplicaciones críticas.

Métodos

Para la medición del tiempo de ejecución y la evaluación del rendimiento del sistema paralelo, se integró una ESP32 programada en MicroPython. Esta ESP32 registraba los tiempos de inicio y finalización de las operaciones en las FPGAs, permitiendo una evaluación precisa del tiempo total de procesamiento. La comunicación entre la ESP32 y las FPGAs se llevó a cabo mediante el protocolo CAN-Bus, donde las señales indicaban a la ESP32 el inicio y la finalización del procesamiento en cada FPGA. El código de la ESP32 capturaba estos eventos y convertía los tiempos registrados en milisegundos, facilitando su análisis.

La elección de la ESP32 se basó en su versatilidad y capacidad para gestionar múltiples entradas y salidas, así como su compatibilidad con el protocolo CAN-Bus, lo que permitió implementar la medición sin afectar el rendimiento del sistema de FPGAs. La programación y simulación de las FPGAs se realizaron en el entorno Quartus II, utilizando VHDL para desarrollar y verificar los códigos (“Altera Cyclone IV EP4CE15F23C8N”, s/f; Altera, 2014; www.alldatasheet.com, s/f). Se realizaron simulaciones funcionales para comprobar que la lectura de la matriz se ejecutaba correctamente en paralelo y que los tiempos medidos por la ESP32 eran coherentes con las expectativas de rendimiento.

El entorno experimental incluyó la evaluación de las frecuencias de operación de los dispositivos y la medición de los tiempos de ejecución. Los parámetros clave medidos fueron el tiempo total

 ISSN: 2594-1410

de procesamiento de la matriz de 999x999 elementos y la precisión en la sincronización entre las FPGAs. Finalmente, los resultados obtenidos en los displays de 7 segmentos y los tiempos medidos por la ESP32 fueron comparados con las simulaciones previas, confirmando una reducción del 50% en los tiempos de procesamiento esperados.

Diseño de la Matriz en la FPGA

La implementación principal del proyecto se centró en el procesamiento de una matriz de 999x999 elementos utilizando dos FPGAs Cyclone IV EP4CE15F23C8N. El diseño fue elaborado en VHDL, un lenguaje de descripción de hardware que permitió crear un módulo para que cada FPGA leyera y procesara su respectiva parte de la matriz de manera eficiente. El objetivo fue dividir el trabajo de tal forma que cada FPGA procesara 499,500 elementos, lo que mejoró el tiempo de lectura y procesamiento total.

Paralelismo entre las FPGAs

El procesamiento paralelo entre las FPGAs fue clave para reducir el tiempo de operación. Se diseñaron señales de control que permitían sincronizar ambas FPGAs, asegurando que comenzaran la lectura de la matriz al mismo tiempo y que cada una procesara su parte de forma independiente. El reloj compartido fue fundamental para mantener la coherencia de los tiempos y evitar conflictos entre las FPGAs (C. S. et al., 2016; Kastner et al., 2018).

Esquema de los Bloques Lógicos y Sincronización

Cada FPGA fue programada con un bloque lógico encargado de la lectura de la mitad de la matriz. El esquema de bloques incluyó un contador para gestionar el índice de lectura y una serie de multiplexores para controlar las señales de salida hacia los displays de 7 segmentos, que mostraban el progreso en tiempo real. La sincronización entre las FPGAs y el temporizador externo fue lograda mediante la emisión de señales de inicio y fin, enviadas a la ESP32 para medir el tiempo total de operación.

Integración de la ESP32

El siguiente componente crítico del proyecto fue la ESP32, que se utilizó para medir los tiempos de procesamiento en cada FPGA. Esta medición fue esencial para evaluar el impacto del paralelismo y validar los resultados.

Proceso de Medición de Tiempos mediante la ESP32

La ESP32 se programó en MicroPython, (Bell, 2020) para recibir señales de inicio y fin desde las FPGAs, a través del protocolo CAN-Bus. Estas señales activaban un temporizador en la ESP32, lo que permitió medir con precisión el tiempo total que tardaba cada FPGA en completar su parte de la matriz. Este método externo de medición fue necesario para no interferir con los procesos internos de las FPGAs.

Hardware y Software Utilizado para la Conexión

El hardware de la ESP32 fue seleccionado por su capacidad para manejar múltiples señales y su compatibilidad con el protocolo CAN-Bus (Nikolov & Gotseva, 2024; Pham et al., 2022), lo que facilitó la transmisión de datos entre las FPGAs y la ESP32. La programación se realizó utilizando MicroPython, y el código incluyó rutinas para capturar las señales en tiempo real y convertirlas en una métrica precisa del tiempo de ejecución.

Entorno de Simulación y Pruebas

Desarrollo

El desarrollo de los módulos de VHDL se realizó en el entorno Quartus II, una herramienta de programación y simulación de FPGAs (H. Shen et al., 2011; Yang et al., 2014). Este software permitió realizar simulaciones funcionales antes de la síntesis del hardware, garantizando que los módulos de procesamiento y control funcionaran correctamente antes de ser implementados físicamente en las FPGAs.

Configuración Experimental

La FPGA Cyclone IV EP4CE15F23C8N utilizada en este proyecto ofrece una alta capacidad de procesamiento con 15,408 elementos lógicos y varios bloques de memoria RAM integrados, lo que la hizo adecuada para manejar la matriz de 999x999 elementos. Las pruebas experimentales se realizaron bajo una frecuencia de operación de 50 MHz, lo que permitió gestionar grandes volúmenes de datos en tiempos razonables.

Proceso de Validación de Resultados

Para validar los resultados, se llevaron a cabo pruebas comparativas entre el procesamiento de la matriz con una sola FPGA y con las dos FPGAs en paralelo. Las mediciones realizadas por la ESP32, como se muestran lo tiempo finales en la tabla 1, demostraron que el sistema de procesamiento paralelo redujo el tiempo total de procesamiento en un 50%, lo que confirmó la efectividad del enfoque implementado.

Este entorno de simulación y pruebas aseguró que el diseño cumpliera con los requisitos de rendimiento, proporcionando una base sólida para su aplicación en proyectos más complejos.

Resultados

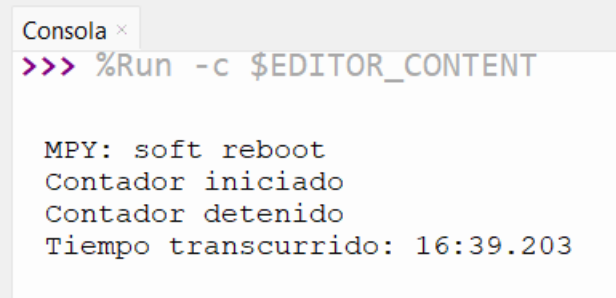
Comparación de los Tiempos de Procesamiento Medidos por la ESP32

Para evaluar la mejora en el rendimiento del sistema de procesamiento paralelo, se midieron los tiempos de operación utilizando una sola FPGA y luego dos FPGAs en paralelo. Las mediciones se realizaron con una ESP32 conectada mediante el protocolo CAN-Bus, que registró los tiempos de inicio y fin de cada operación.

La Figura 1 muestra el tiempo que le tomo a una FPGA, el llenado de una matriz de 999x 999 que son alrededor de 998001 elementos, en la cual se tardó un tiempo total de 16 minutos con 39 segundos y 203 milésimas de segundo el llenado de esta matriz cuadrada.

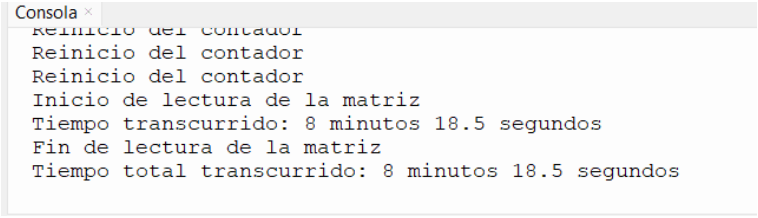
 ISSN: 2594-1410

Figura 1. *Medición de tiempo transcurrido por una FPGA en la matriz 999x999*



Por otro lado, la Figura 2, muestra el mismo proceso, pero con el proceso paralelo, de dos FPGA`s, con la misma matriz cuadrada con un tiempo de 8 minutos 18 segundos y 500 milisegundos.

Figura 2. *Medición de tiempo transcurrido por dos FPGAs en paralelo en la matriz 999x999*



Los resultados de estas mediciones, de los dos procesos son presentados en la tabla 1, en donde se puede ver la comparación con el proceso paralelo y sin el proceso paralelo, como se puede mostrar en reducción de tiempo represento en un 50%.

Tabla 1.

Comparación de resultados entre tiempos.

Operación	Tiempo con 1 FPGA (Minutos)	Tiempo con 2 FPGA (Minutos)	Reducción %
Lectura de una matriz 999*999	16.39	8.18	50%

El uso de dos FPGAs redujo el tiempo de procesamiento a la mitad, lo que confirma la efectividad del paralelismo implementado. La ESP32 permitió validar que ambas FPGAs trabajaban sincronizadas, cada una procesando la mitad de la matriz sin interferencias ni retrasos.

Evaluación del Uso de Recursos

Además del análisis de tiempos, se evaluó el uso de recursos de las FPGAs Cyclone IV en términos de memoria utilizada, LUTs (Look-Up Tables) y BRAM (Block RAM), tanto para la implementación con una sola FPGA como con dos FPGAs en paralelo.

Memoria Utilizada:

- Memoria total disponible: 516 Kbits (disponibles en las dos FPGAs combinadas).
- Memoria utilizada con una FPGA: 60% de la memoria total disponible.

- Memoria utilizada con dos FPGAs: 35% en cada FPGA, lo que optimizó el uso del recurso de memoria al distribuir la carga.

Uso de LUTs:

- Uso con una FPGA: Se utilizaron aproximadamente 8,000 LUTs (de un total de 15,408 disponibles en la Cyclone IV).
- Uso con dos FPGAs: Cada FPGA utilizó 4,000 LUTs, lo que permitió un uso más eficiente de los recursos sin sobrecargar ninguna de las FPGAs.

BRAM:

- Uso de BRAM con una FPGA: Se ocupó un 75% de los bloques de memoria BRAM disponibles.
- Uso de BRAM con dos FPGAs: Al dividir el procesamiento, se utilizó un 40% de BRAM en cada FPGA, reduciendo la carga en los bloques de memoria.

Este reparto de recursos no solo permitió reducir los tiempos de procesamiento, sino que también optimizó el uso de los recursos internos de las FPGAs, evitando la saturación de una sola FPGA y permitiendo un equilibrio en el uso de LUTs y BRAM. Estos resultados demuestran que el paralelismo entre las dos FPGAs no solo mejora el rendimiento en términos de tiempo, sino que también aprovecha mejor los recursos de hardware disponibles.

Esta estrategia demostró ser sumamente eficaz en aplicaciones de control automotriz y en otros sistemas que requieren procesamiento en tiempo real, donde la reducción en el tiempo de respuesta es crítica para el rendimiento general del sistema. Además, el uso de la ESP32 como un temporizador externo permitió una evaluación precisa del rendimiento del sistema, confirmando que la arquitectura propuesta es capaz de manejar operaciones más complejas y críticas en futuras aplicaciones.

Discusión

Los resultados obtenidos en este proyecto confirman que la implementación de un sistema de procesamiento paralelo con dos FPGAs puede mejorar significativamente el rendimiento de tareas intensivas en cálculo, como la lectura de una matriz de 999x999 elementos. El tiempo de procesamiento se redujo en un 50%, lo que valida la eficacia del enfoque de dividir la carga de trabajo entre múltiples FPGAs. Este resultado es crucial para aplicaciones que requieren alto rendimiento en tiempo real, como los sistemas de control automotriz, donde cada milisegundo es crítico para la toma de decisiones en tiempo real.

En comparación con investigaciones previas sobre el uso de FPGAs en sistemas de control, nuestros resultados están alineados con estudios que destacan la capacidad de las FPGAs para ejecutar operaciones paralelas y mejorar la eficiencia de sistemas embebidos. Otros trabajos han demostrado que la implementación de arquitecturas paralelas en hardware reconfigurable optimiza el uso de los recursos y acelera significativamente los tiempos de procesamiento.

Uno de los principales desafíos en la implementación de este sistema fue garantizar una sincronización precisa entre las dos FPGAs y la ESP32. La correcta transmisión de señales de control y el manejo de tiempos de inicio y fin fue crucial para asegurar que ambas FPGAs trabajaran de manera coordinada. Aunque la ESP32 funcionó como temporizador externo, su

integración requirió múltiples ajustes para asegurar que no hubiera desincronización, lo cual podría haber afectado los resultados de rendimiento.

Adicionalmente, aunque el procesamiento paralelo demostró ser efectivo, la complejidad del sistema aumentó al incluir más componentes y señales a sincronizar. Esto presenta un área de oportunidad para futuras mejoras en la eficiencia y simplicidad del diseño, especialmente al considerar aplicaciones más complejas que podrían incluir más FPGAs o procesos adicionales.

Los conocimientos adquiridos en este estudio no solo optimizaron el procesamiento paralelo, sino que también proporcionaron las bases para futuros desarrollos. Uno de los próximos pasos clave será la implementación de una ecuación de batería en las FPGAs, con el objetivo de simular el comportamiento de una batería de automóvil. Esta simulación incluirá el modelado del ciclo de vida de la batería, los procesos de carga y descarga, y el estado de salud de esta en tiempo real.

Este proyecto futuro tiene implicaciones importantes para aplicaciones automotrices, especialmente en el campo de los vehículos eléctricos, donde la gestión eficiente de la batería es fundamental para la operación del vehículo. El enfoque en procesamiento paralelo permitirá que las FPGAs manejen de manera eficaz los cálculos complejos asociados con la simulación de la batería, lo que podría resultar en modelos más precisos y un mejor control sobre los sistemas de energía del automóvil.

El conocimiento adquirido en este trabajo de investigación será crucial para desarrollar un sistema robusto y eficiente que pueda simular una batería automotriz en tiempo real, mejorando la capacidad de control y monitoreo de sistemas eléctricos en el sector automotriz.

Conclusiones

Este estudio ha demostrado que la implementación de un sistema de procesamiento paralelo utilizando dos FPGAs puede reducir significativamente el tiempo de procesamiento en aplicaciones intensivas en cálculo, como la lectura de una matriz de 999x999 elementos. Al dividir la carga de trabajo entre ambas FPGAs, el tiempo total de procesamiento se redujo en un 50%, pasando de 16.39 minutos a 8.18 minutos. Además, la integración de una ESP32 para medir tiempos en tiempo real permitió validar la eficiencia del sistema, asegurando que las FPGAs estaban correctamente sincronizadas y coordinadas durante todo el proceso.

El diseño presentado no solo resultó ser efectivo en términos de procesamiento paralelo, sino que también destacó la viabilidad de utilizar una ESP32 como temporizador externo para monitorear y medir el rendimiento del sistema. La combinación de ambos dispositivos permitió una mejora en el control y la medición de las operaciones, lo que refuerza la utilidad de las FPGAs en aplicaciones de control automotriz y otros sistemas embebidos que requieren procesamiento en tiempo real.

El uso de la ESP32 como controlador externo fue fundamental para garantizar la precisión en la medición de tiempos y evitar cualquier interferencia con las operaciones internas de las FPGAs. Esto sugiere que la ESP32 puede ser utilizada en futuras implementaciones de sistemas paralelos para monitorear y gestionar tareas complejas de manera eficiente.

Los resultados de esta investigación tienen importantes implicaciones para el desarrollo de sistemas embebidos y el procesamiento paralelo en hardware reconfigurable. El enfoque presentado puede ser aplicado a diversas áreas que requieren un procesamiento rápido y eficiente, como los sistemas de control en tiempo real para automóviles, robótica avanzada, y dispositivos de alta demanda de cálculo.

Además, los conceptos aprendidos en este proyecto servirán como base para la futura implementación de una ecuación de batería en una FPGA, lo que permitirá simular el comportamiento de una batería automotriz en tiempo real. Esta simulación abrirá nuevas posibilidades para la gestión energética en vehículos eléctricos y sistemas similares, donde el control preciso del ciclo de vida de la batería es crucial.

Para futuros trabajos, se recomienda explorar la posibilidad de ampliar el sistema paralelo con más FPGAs o integrar otros dispositivos embebidos para aumentar la capacidad de procesamiento y optimizar aún más el rendimiento. También sería beneficioso investigar nuevas técnicas de sincronización y comunicación entre FPGAs y otros dispositivos externos, con el fin de reducir la complejidad del sistema y mejorar su escalabilidad.

En aplicaciones industriales, la metodología presentada puede ser aplicada en sistemas que requieran procesamiento de datos en tiempo real a gran escala, como en la manufactura inteligente, redes de sensores y sistemas de control automotriz. Asimismo, la simulación de la batería en una FPGA podría extenderse para optimizar el diseño de sistemas de gestión de energía en diversas industrias.

Referencias

Adil Yazdeen, A., Zeebaree, S. R. M., Mohammed Sadeeq, M., Kak, S. F., Ahmed, O. M., & Zebari, R. R. (2021). FPGA Implementations for Data Encryption and Decryption via Concurrent and Parallel Computation: A Review. *Qubahan Academic Journal*, *1*(2), 8–16. https://doi.org/10.48161/qaj.v1n2a38

Altera Cyclone IV EP4CE15F23C8N. (s/f). *FPGA Cookbook*. Recuperado el 11 de octubre de 2024, de https://www.fpga-cookbook.com/altera-development-boards/altery-cyclone-iv-ep4ce15f23c8n/

Altera, I. N. C. (2014). Cyclone IV Device Handbook. En *2010-12-01)[2012-05-16]. Http://www. Altera, com*.

Bell, C. (2020). Programming in MicroPython. En C. Bell (Ed.), *Beginning Sensor Networks with XBee, Raspberry Pi, and Arduino: Sensing the World with Python and MicroPython* (pp. 103–142). Apress. https://doi.org/10.1007/978-1-4842-5796-8_3

Bosse, S. (2022). VNetOS: Virtualised Distributed and Parallel Sensor Network Operating Environment for the IoT and SHM. *Engineering Proceedings*, *27*(1), Article 1. https://doi.org/10.3390/ecsa-9-13212

C. S., A. K., S., P. B., & Kumar, K. J. (2016). Design and Development of High-speed Data Acquisition System with Cyclone FPGA. *Proceedings of the 7th International Conference on Computing Communication and Networking Technologies*, 1–5. https://doi.org/10.1145/2967878.2967896

Cabanes, Q., Senouci, B., & Ramdane-Cherif, A. (2019). A Complete Multi-CPU/FPGA-based Design and Prototyping Methodology for Autonomous Vehicles: Multiple Object Detection and Recognition Case Study. *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, 158–163. https://doi.org/10.1109/ICAIIC.2019.8669047

Cao, Y., Guo, S., Jiang, S., Zhou, X., Wang, X., Luo, Y., Yu, Z., Zhang, Z., & Deng, Y. (2022). Parallel Optimisation and Implementation of a Real-Time Back Projection (BP) Algorithm for SAR Based on FPGA. *Sensors*, *22*(6), 2292. https://doi.org/10.3390/s22062292

Cesarano, G. (2018). *FPGA Implementation of a Deep Learning Inference Accelerator for Autonomous vehicles* [Laurea, Politecnico di Torino]. https://webthesis.biblio.polito.it/9033/

Dias, L. A., Ferreira, J. C., & Fernandes, M. A. C. (2020). Parallel Implementation of K-Means Algorithm on FPGA. *IEEE Access*, *8*, 41071–41084. https://doi.org/10.1109/ACCESS.2020.2976900

Ghielmetti, N., Loncar, V., Pierini, M., Roed, M., Summers, S., Aarrestad, T., Petersson, C., Linander, H., Ngadiuba, J., Lin, K., & Harris, P. (2022). Real-time semantic segmentation on FPGAs for autonomous vehicles with hls4ml. *Machine Learning: Science and Technology*, *3*(4), 045011. https://doi.org/10.1088/2632-2153/ac9cb5

Guo, K., Sui, L., Qiu, J., Yao, S., Han, S., Wang, Y., & Yang, H. (2016). From model to FPGA: Software-hardware co-design for efficient neural network acceleration. *2016 IEEE Hot Chips 28 Symposium (HCS)*, 1–27. https://doi.org/10.1109/HOTCHIPS.2016.7936208

Hao, C., Sarwari, A., Jin, Z., Abu-Haimed, H., Sew, D., Li, Y., Liu, X., Wu, B., Fu, D., Gu, J., & Chen, D. (2019). A Hybrid GPU + FPGA System Design for Autonomous Driving Cars. *2019 IEEE International Workshop on Signal Processing Systems (SiPS)*, 121–126. https://doi.org/10.1109/SiPS47522.2019.9020540

Hasegawa, K., Takasaki, K., Nishizawa, M., Ishikawa, R., Kawamura, K., & Togawa, N. (2019). Implementation of a ROS-Based Autonomous Vehicle on an FPGA Board. *2019 International Conference on Field-Programmable Technology (ICFPT)*, 457–460. https://doi.org/10.1109/ICFPT47387.2019.00092

Hronek, P. (2023). *CAN Bus Latency Test Automation for Continuous Testing and Evaluation*. https://support.dce.felk.cvut.cz/mediawiki/images/0/0a/Bp_2023_hronek_pavel.pdf

Huang, Z., Yu, Q., & Dong, H. (2024). The study of remote online upgrade method for FPGA based on CAN bus. *International Conference on Optoelectronic Information and Computer Engineering (OICE 2024)*, *13255*, 55–60. https://doi.org/10.1117/12.3040242

Humaidi, A. J., Hasan, S., & Fadhel, M. A. (2018). FPGA-Based Lane-Detection Architecture for autonomous vehicles: A real-time design and development. *ASIA LIFE SCIENCES*.

Kasem, A., Reda, A., Vásárhelyi, J., & Bouzid, A. (2021). A Survey about Intelligent Solutions for Autonomous Vehicles based on FPGA. *Carpathian Journal of Electronic & Computer Engineering*, *13*(2). https://search.ebscohost.com/login.aspx?direct=true&profile=ehost&scope=site&authtype=crawler&jrnl=18449689&AN=148608132&h=CUXRsUGA8MDVO6GH9l5prTUl%2BwO%2F53T4xcqsQO3r4wJ7C0y5J7382tlEyerpcB6cVo7o0ikBVEJxYIZo1cJmSg%3D%3D&crl=c

Kastner, R., Matai, J., & Neuendorffer, S. (2018). *Parallel Programming for FPGAs* (No. arXiv:1805.03648). arXiv. https://doi.org/10.48550/arXiv.1805.03648

Kojima, A., & Nose, Y. (2018). Development of an Autonomous Driving Robot Car Using FPGA. *2018 International Conference on Field-Programmable Technology (FPT)*, 411–414. https://doi.org/10.1109/FPT.2018.00087

Li, Y., Li, S. E., Jia, X., Zeng, S., & Wang, Y. (2022). FPGA accelerated model predictive control for autonomous driving. *Journal of Intelligent and Connected Vehicles*, *5*(2), 63–71. Journal of Intelligent and Connected Vehicles. https://doi.org/10.1108/JICV-03-2021-0002

Mueller, R., Teubner, J., & Alonso, G. (2009). Data processing on FPGAs. *Proc. VLDB Endow.*, *2*(1), 910–921. https://doi.org/10.14778/1687627.1687730

Nesbø, S. V., Alme, J., Bonora, M., Ersdal, M. R., Giubilato, P., Helstrup, H., Lupi, M., Rinella, G. A., Røhrich, D., & Schambach, J. (2020). *Implementation of a CAN bus interface for the Detector Control System in the ALICE ITS Upgrade*. https://bora.uib.no/bora-xmlui/handle/11250/2758211

Nikolov, N., & Gotseva, D. (2024). Make a Prototype of IoT Connected Diagnostic Tool Using Esp32 and MQTT for Reading Data from Car CAN Bus OBD2. *2024 59th International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST)*, 1–4. https://doi.org/10.1109/ICEST62335.2024.10639614

Nur-A-Alam, Ahsan, M., Based, M. A., Haider, J., & Rodrigues, E. M. G. (2021). Smart Monitoring and Controlling of Appliances Using LoRa Based IoT System. *Designs*, *5*(1), Article 1. https://doi.org/10.3390/designs5010017

Pawar, R. (2022). *Implementation of CAN on FPGA for Security Evaluation Purpose*. https://www.academia.edu/download/95477386/IRJET_V9I1188.pdf

Pham, N. N., Leuchter, J., Pham, K. L., & Dong, Q. H. (2022). Battery Management System for Unmanned Electric Vehicles with CAN BUS and Internet of Things. *Vehicles*, *4*(3), Article 3. https://doi.org/10.3390/vehicles4030037

Rodríguez, A., Navarro, A., Asenjo, R., Corbera, F., Gran, R., Suárez, D., & Nunez-Yanez, J. (2020). Parallel multiprocessing and scheduling on the heterogeneous Xeon+FPGA platform. *The Journal of Supercomputing*, *76*(6), 4645–4665. https://doi.org/10.1007/s11227-019-02935-1

Seng, K. P., Lee, P. J., & Ang, L. M. (2021). Embedded Intelligence on FPGA: Survey, Applications and Challenges. *Electronics*, *10*(8), 895. https://doi.org/10.3390/electronics10080895

Shen, H., Feng, Y., & Tan, G. (2011). *Research on time synchronization of Class B LXI bus*. 112–116. https://doi.org/10.1049/cp.2011.0858

Shen, M., Luo, G., & Xiao, N. (2021). Combining Static and Dynamic Load Balance in Parallel Routing for FPGAs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *40*(9), 1850–1863. https://doi.org/10.1109/TCAD.2020.3031259

Shen, M., & Xiao, N. (2020). Towards Serial-Equivalent Multi-Core Parallel Routing for FPGAs. *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 1139–1144. https://doi.org/10.23919/DATE48585.2020.9116313

Shimiao, C., Dashuang, Y., Shuyan, N., & Ke, Z. (2020). Design and Implementation of Microsatellite CAN Bus Communication. *2020 7th International Conference on*

Information Science and Control Engineering (ICISCE), 2354–2360. https://ieeexplore.ieee.org/abstract/document/9532114/

Stojilović, M. (2017). Parallel FPGA routing: Survey and challenges. *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, 1–8. https://doi.org/10.23919/FPL.2017.8056782

Wei, K., Honda, K., & Amano, H. (2018). FPGA Design for Autonomous Vehicle Driving Using Binarized Neural Networks. *2018 International Conference on Field-Programmable Technology (FPT)*, 425–428. https://doi.org/10.1109/FPT.2018.00091

www.alldatasheet.com. (s/f). *EP4CE15F23C8N datasheet(4/14 Pages) ALTERA*. Recuperado el 11 de octubre de 2024, de http://www.alldatasheet.com/html-pdf/536408/ALTERA/EP4CE15F23C8N/906/4/EP4CE15F23C8N.html

Yang, H. J., Fan, H., & Dong, H. G. (2014). Design and Implementation of a RISC Processor on FPGA. *Advanced Materials Research*, *981*, 58–61. https://doi.org/10.4028/www.scientific.net/AMR.981.58

Zhang, Y. Y., Zhang, L., Shang, Z. Q., Su, Y. R., Wu, Z., & Yan, F. B. (2022). A New Multichannel Parallel Real-time FFT Algorithm for a Solar Radio Observation System Based on FPGA. *Publications of the Astronomical Society of the Pacific*, *134*(1033), 034502. https://doi.org/10.1088/1538-3873/ac5212

