Revista Ciencia Aplicada
Conocimiento Multidisciplinario ISSN: 3122-3664

Artículo Científico

Comparación y estudio del desempeño de dos sistemas embebidos en FPGA

Abimael Jiménez[1], Gehová López-González[2], Omar Aguilar Loreto[3]

[1]Departamento de Ingeniería Eléctrica y Computación de la Universidad Autónoma de Ciudad Juárez, Av. De Charro 450, Juárez, Chihuahua México
[2,3]Departamento de Ingeniería, Universidad de Guadalajara, Av. Revolución 151, Autlán, Jalisco, México

Resumen: Los sistemas embebidos con FPGA (Field Programmable Gate Arrays) son utilizados en muchas aplicaciones en áreas de comunicaciones, automotriz, biomédica y aeroespacial debido a sus ventajas de reconfiguración de hardware y paralelismo. En este trabajo se presenta un análisis de desempeño de dos sistemas embebidos implementados en la tarjeta Basys 3, la cual contiene un FPGA Artix-7. En ambos sistemas se desarrolló el software, el cual integra una aplicación y controladores de los módulos de hardware. En la parte de hardware un diseño fue implementado con módulos de propiedad intelectual y otro con módulos desarrollados en lenguaje de descripción de hardware. Este último presentó el mejor desempeño al utilizar solo el 3.37% de los bloques lógicos disponibles del FPGA Artix-7 y un consumo de potencia de 0.106 W.

Palabras clave: sistema embebido, VHDL, diseño de harware, desarrollo de software, desempeño.

1. Introducción

En la actualidad, los sistemas embebidos en FPGA (Field Programmable Gate Arrays) representan una solución eficiente y flexible para aplicaciones que requieren procesamiento en tiempo real, bajo consumo de energía y personalización a nivel de hardware. Estas características proporcionan una solución intermedia entre los ASICs (Application-Specific Integrated Circuits) y los procesadores de propósito general.

Gracias a la integración de procesadores soft como Miicrobalaze [1] y Nios V [2], implementados en HDL (Hardware Description Language), y procesadores físicos como ARM Cortex, desarrollado en nodos tecnológicos desde los 180 nm [3]; es posible el diseño de sistemas embebidos heterogéneos que combinan lógica programable con procesamiento convencional. Esta arquitectura híbrida ha impulsado su adopción en áreas como visión por computadora, robótica, comunicaciones y sistemas IoT (Internet of Things), donde la capacidad de paralelismo y la adaptabilidad de los FPGAs ofrecen ventajas significativas frente a plataformas tradicionales [4–7].

El diseño de un sistema embebido en FPGA implica el desarrollo de hardware y software. Respecto al hardware, las plataformas de diseño como Vivado [8] y

Primer Volumen

Recibido: mayo 12, 2025
Aceptado: mayo 24, 2025

autor de correspondencia:
abimael.jimenez@uacj.mx

https://doi.org/10.66482/cacxwd03

Quartus [9] contienen diferentes módulos de propiedad intelectual (PI), algunos de uso gratuito y otros a través de licencias. La mayoría de los sistemas embebidos con FPGA integran módulos de PI, simplificación de la etapa diseño. Por otro lado, es posible desarrollar sistemas embebidos sin utilizar módulos de PI. Sin embargo, esto implica un conocimiento avanzado de HDLs como VHDL [10] y Verilog [11]. Por otra parte, el software del sistema embebido se puede implementar en RiscFree IDE y Vitis para FPGAs de Intel y AMD, respectivamente [12,13].

En este trabajo se presenta un estudio del desempeño de sistemas embebidos en FPGA a través del diseño de dos sistemas embebidos mínimos, el primero se diseñó con módulos PI de AMD y en el segundo utiliza módulos de hardware en VHDL. Ambos sistemas embebidos fueron evaluados utilizando la misma aplicación de software y se implementaron en el mismo FPGA.

2. Metodología

Los sistemas embebidos analizados en este trabajo se diseñaron con las herramientas de diseño Vivado Desing Suit y Vitis 2020.2 para el desarrollo de hardware y software, respectivamente [8]. Los dos sistemas se implementaron en la tarjeta de desarrollo Basys 3 de Digilent, la cual contiene el FPGA Artix-7 de AMD [14]. Ambos sistemas utilizaron los módulos de hardware del procesador soft Microblaze, el módulo UART (Universal Asynchronous Receiver-Transmitter) y el módulo GPIO (General Purpose Input/Output). Finalmente, en ambos sistemas se implementaron los drivers correspondientes para cada módulo o periférico y se ejecutó la misma aplicación de software. La única diferencia radica en que el sistema system-ip utilizó módulos de PI de AMD y el sistema system-hdl utilizó módulos de hardware desarrollados en VHDL.

2.1 Diseño de hardware

El diseño *system-ip* se integró con los módulos PI del procesador soft Microblaze, el módulo AXI-UART (Advanced eXtensible Interface - Universal Asynchronous Receiver-Transmitter) y el módulo GPIO como se observa en las Figuras 1 y 2. También se observa que en el diseño aparecen más módulos de hardware, los cuales son agregados automáticamente por la herramienta de Vivado al realizar la automatización de interconexión.

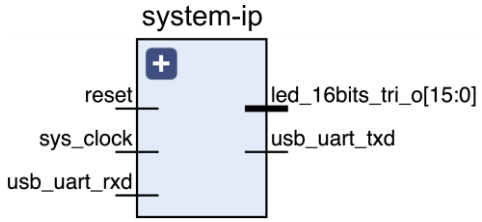

Figura 1. Módulo principal del sistema embebido system-ip implementado en Vivado.

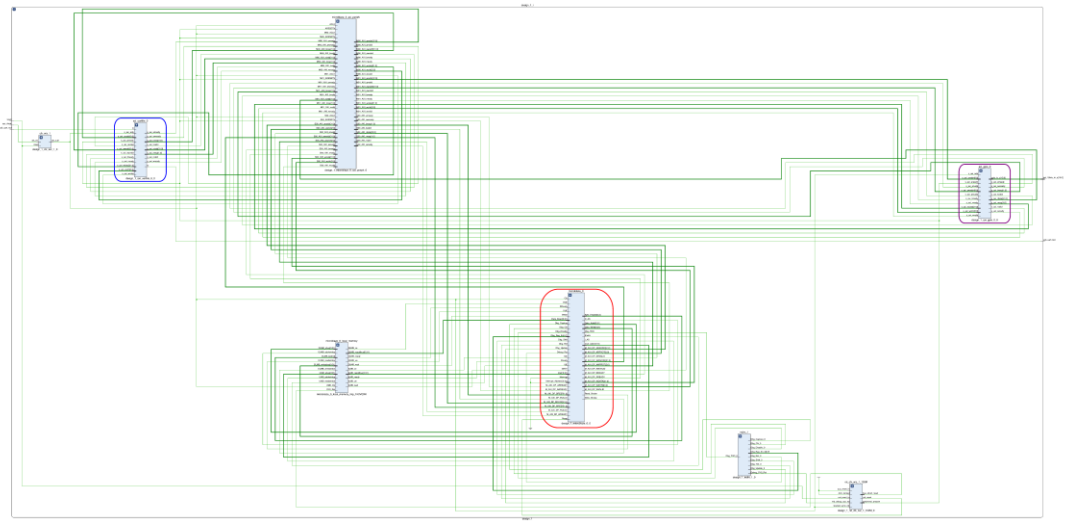

Figura 2. Arquitectura de system-ip en Vivado con los módulos UART, Microblaze y GPIO en recuadros azul, rojo y morado, respectivamente.

En las Figuras 3 y 4 se muestra la arquitectura del diseño *system-hdl*. Este sistema se implementó con el procesador soft Microblaze (CPU en recuadro rojo de la Figura 3), y los módulos UART, GPO, y GPI como se muestra en los recuadros azul, rojo y morado de la Figura 4, respectivamente. Se observa que en este diseño los módulos GPO y GPI están separados en dos módulos en vez de un solo módulo GPIO. Todos los módulos del diseño *system-hdl*, a excepción del procesador, están desarrollados en VHDL y se tomaron del sistema Vanilla [14]. En ambos sistemas el procesador Microblaze se configuró con 128 kb de memoria RAM y una frecuencia de reloj de 100 MHz.

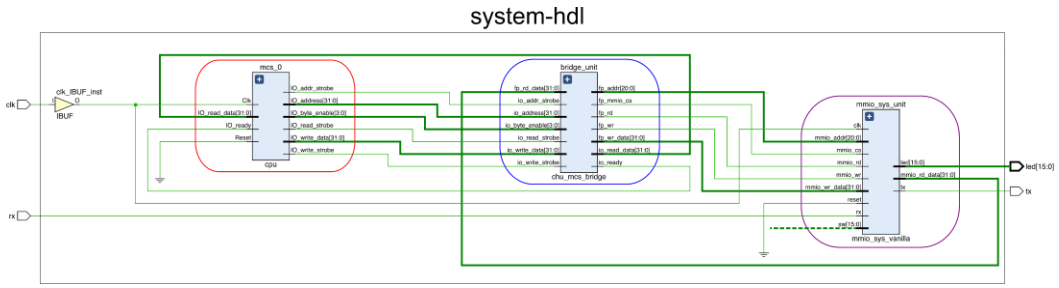

Figura 3. Módulo principal del sistema embebido system-hdl con los módulos chu_mcs_bridge, cpu (Microblaze) y mmio_sys_vanilla en recuadros azul, rojo y morado, respectivamente.

Después de agregar las fuentes de los módulos de hardware de cada sistema y su respectivo archivo de distribución de pines de la tarjeta Basys 3, se realizó el proceso de síntesis y generación del archivo .bit (bitstream) para programar el hardware en el FPGA Artix-7. Finalmente, se exportó el hardware de ambos sistemas, archivo .xsa (Xilinx Support Archive) que describe la plataforma de hardware.

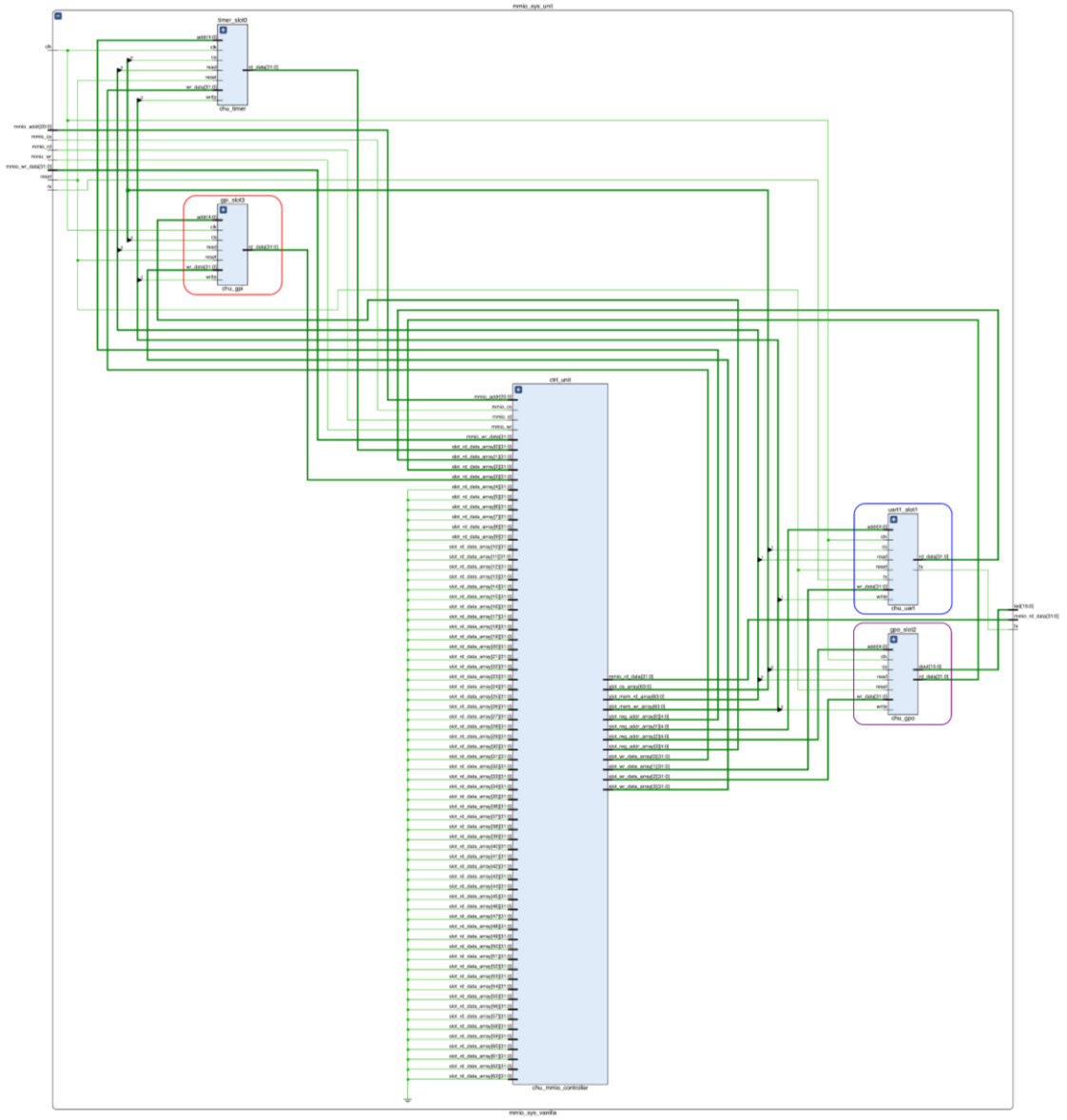

Figura 4. Arquitectura de system-hdl con los módulos UART, GPI y GPO en recuadros azul, rojo y morado, respectivamente.

2.2 Diseño de software

El diseño de software se realizó de acuerdo con el diagrama de flujo de la Figura 5a. El primer paso fue la creación de un espacio de trabajo en Vitis. Posteriormente, se creó la plataforma, importando la plataforma de hardware (archivo .xsa) que se exportó en Vivado; la cual es una combinación de componentes de hardware y componentes de software (dominios/Board Support Package (BSP), etc.). A continuación, se generó el BSP, que proporciona el soporte básico para la plataforma

de hardware, facilitando la inicialización del sistema al encenderse y permitiendo la ejecución del programa de aplicación.

En ambos sistemas se empleó un BSP tipo autónomo que ofrece una capa de software sencilla y de bajo nivel. Esta configuración brinda acceso directo a las funciones básicas del procesador sin necesidad de un sistema operativo, como se observa en la Figura 5a.

Después, se implementaron los controladores para los módulos de hardware UART, GPO y GPI como se observa en la Figura 5b. En el diseño system-ip, estos controladores se desarrollaron a partir de las bibliotecas disponibles para cada módulo de PI. En cambio, en el diseño system-hdl, se utilizaron los controladores desarrollados en el sistema Vanilla [14].

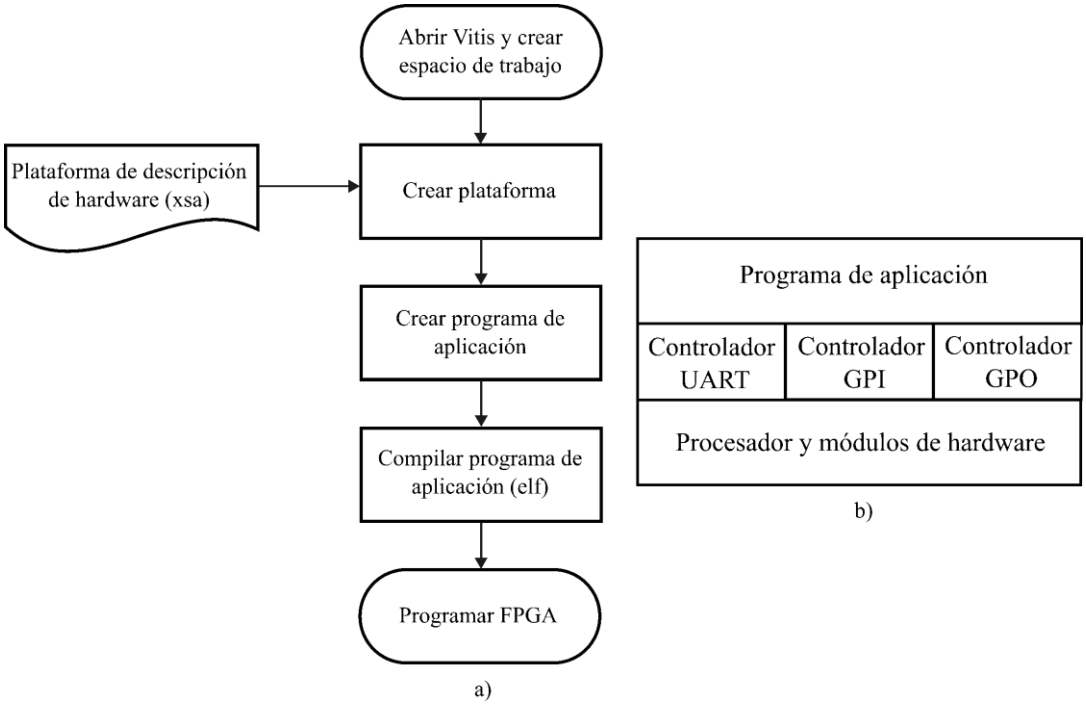

Figura 5. a) Diagrama de flujo para desarrollar el software de los diseños. b) Capas que integran el software.

Posteriormente, se desarrolló el programa de aplicación en C++ que contiene un ciclo infinito con el que inicia el sistema, interactuando con cada módulo de hardware y el procesador. Las rutinas que incluye la aplicación son el parpadeo de todos los LEDs cinco veces cada segundo, el encendido de los LEDs uno por uno cada segundo y el envío de mensajes a través de UART. Tras ello, el software se compiló en Vitis, generando el archivo ejecutable (.elf), el cual se carga en la memoria RAM del procesador MicroBlaze de ambos diseños. Esta carga se realizó desde Vivado, asociando el archivo (.elf) con la arquitectura de hardware correspondiente.

Finalmente, se regeneró el bitstream en Vivado, integrando tanto el hardware como el software, para programar el FPGA Artix-7 como se observa en la Figura 5a. Para verificar el correcto funcionamiento de ambos sistemas, se utilizó una conexión serial mediante la herramienta PuTTY, lo que permitió visualizar mensajes, así como depurar y optimizar el software.

3. Resultados

En esta sección se presentan los resultados obtenidos de la implementación de ambos diseños y se realiza un análisis comparativo del desempeño y los recursos utilizados.

En la Figura 6 se puede observar que el diseño system-ip utiliza la mayor cantidad de recursos. El diseño utilizó 6.96 % de los CLBs (Configurable Logic Blocks) disponibles en el FPGA Artix 7 donde la mayoría de los elementos fueron LUTs (Look-Up Tables); y principalmente, se utilizaron para el diseño de circuitos lógicos con un 6.38 % y solo el 1.44 % de las LUTS se utilizó como memoria. Por su parte el diseño system-hdl utiliza la menor cantidad de recursos, a excepción de los bloques IOB (Input/Output Block) y multiplexores. Esto se debe a que el sistema Vanilla [14] está preconfigurado para integrar hasta 64 módulos de hardware como se observa en la Figura 4. En los registros de estos módulos, el procesador puede escribir y leer datos a través de un decodificar 6 a 64 y un multiplexor 64 a 1, respectivamente.

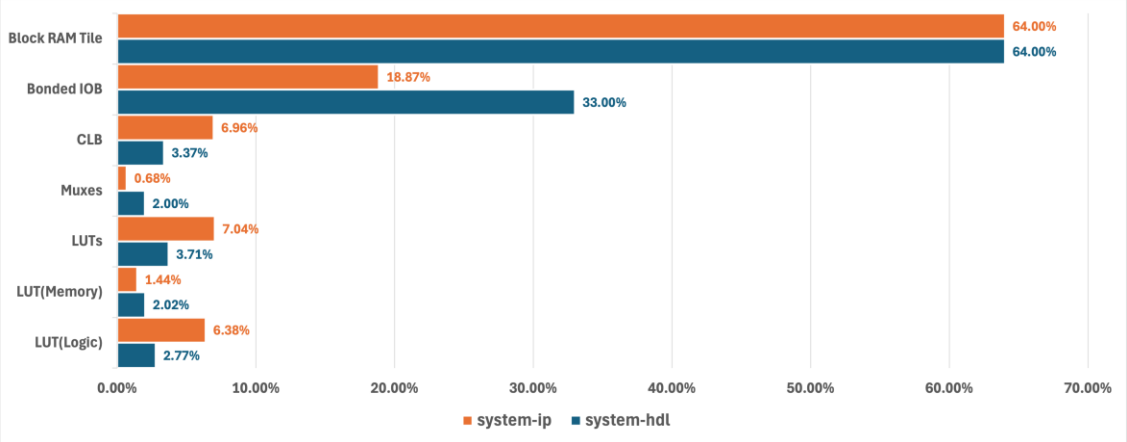

Figura 6. Comparación de los recursos de hardware utilizados en los diseños system-ip y system hdl.

En la Figura 7 se muestra el consumo de potencia de ambos sistemas. El consumo de potencia total de cada diseño está definido por

$$P = P_{static} + P_{dynamic} \quad (1)$$

donde P_{static} representa el consumo de potencia estática y $P_{dynamic}$ es la potencia dinámica. Las dos contribuciones más importantes de la potencia estática son la potencia consumida cuando los transistores están completamente apagados o

encendidos y la debida a corrientes de fuga. La potencia dinámica tiene tres contribuciones importantes, la frecuencia del sistema, la capacitancia de carga y el voltaje de alimentación; por lo tanto, está relacionada con la velocidad de conmutación (encendido / apagado) de los transistores.

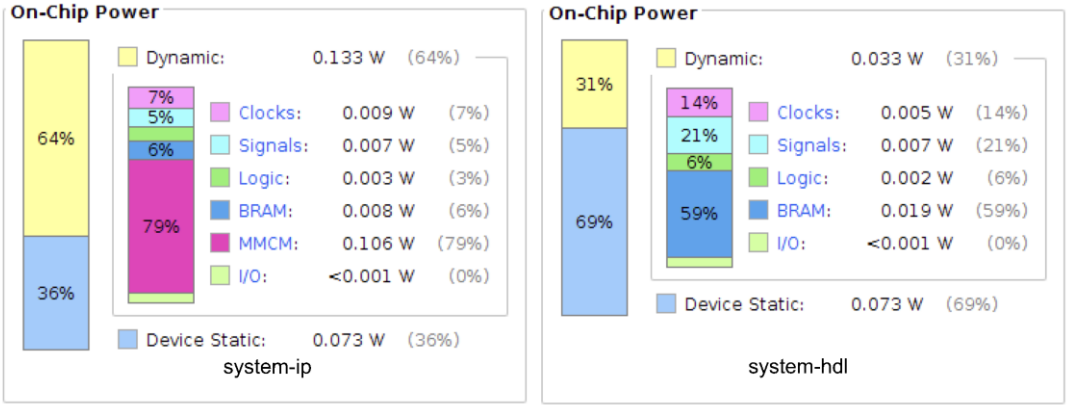

Figura 7. Comparación del consumo de potencia en los diseños system-ip y system hdl.

En la Figura 7 se observa que el consumo de potencia total para el diseño system-ip fue de 0.206 W; mientras que, para el diseño system-hdl fue de 0.106 W. También se observa que el consumo de potencia dinámico es mayor en el diseño system-ip con 0.133 W contra solo 0.033 W en el diseño system-hdl. El 79 % de la potencia dinámica del diseño system-ip se consume en el módulo MMCM (Mixed-Mode Clock Manager). Este módulo es una fuente de señales de reloj para FPGAs y sistemas embebidos de AMD que se utiliza para generar múltiples señales de reloj con varias relaciones de frecuencia y fase a partir de una sola señal de reloj de entrada. Este módulo es principalmente utilizado por el protocolo AXI (Advanced eXtensible Interface).

4. Conclusiones

Los diseños system-ip y system-hdl se desarrollaron satisfactoriamente. El diseño system-hdl utilizó la menor cantidad de recursos de un FPGA Artix-7 y el menor consumo de potencia. Esto plantea la posibilidad de desarrollar sistemas embebidos más complejos en FPGAs más económicos y pequeños; utilizando módulos de hardware en VHDL en lugar de módulos de PI de AMD. Como trabajo a futuro se continuarán agregando módulos de hardware a estos sistemas con la finalidad de determinar si esta tendencia se mantiene o cambia al tener sistemas embebidos más grandes y con mayores requerimientos.

Referencias

1. "AMD MicroBlaze™ Processor", AMD, Accedido: Abr. 25, 2025. [Link]
2. "Nios® V Processor", Intel, Accedido: Abr. 25, 2025. [Link]
3. "Bringing the benefits of Cortex-M processors to FPGA", AMD, Accedido: Abr. 25, 2025. [Link]

4. Magwervyari A and Chen Y., "Review of State-of-the-Art FPGA Applications in IoT Networks". *Sensors*, vol. 22, no. 19, pp. 7496, 2022. https://doi.org/10.3390/s22197496
5. Khan MI and da Silva B., "Harnessing FPGA technology for energy-efficient wearable medical devices". *Electronics*. Vol. 13, no. 20, pp. 4094, 2024. https://doi.org/10.3390/electronics13204094
6. Su, X. and Zuo, G., "Computer Artificial Vision Image Processing System Based on FPGA", en International Conference on Cognitive based Information Processing and Applications (CIPA 2021), J. Jansen, B., Liang, H., Ye, J., Eds., Springer, Singapore, cap. 85. pp. 194-201, 2021. https://doi.org/10.1007/978-981-16-5854-9_24
7. Machado, F., Nieto, R., Fernández-Conde, J., Lobato, D. and José M. Cañas, "Vision-based robotics using open FPGAs", *Microprocessors and Microsystems*, Vol. 103, pp. 104974, 2023. https://doi.org/10.1016/j.micpro.2023.104974
8. "AMD Vivado™ Design Suite", AMD, Accedido: Abr. 25, 2025. [Link]
9. "Software de diseño Quartus® Prime", Intel, Accedido: Abr. 25, 2025. [Link]
10. "IEEE Standard for VHDL Language Reference Manual", *IEEE*, pp. 1-673, 2019. https://doi.org/10.1109/IEEESTD.2019.8938196
11. IEEE Standard for Verilog Hardware Description Language, *IEEE*, pp. 1-590, 2006. https://doi.org/10.1109/IEEESTD.2006.99495
12. "Nios® V Embedded Processor Design Handbook", Intel, Accedido Mayo 16, 2025. [Link]
13. "Plataforma de software unificada AMD Vitis", AMD, Accedido Mayo 16, 2025. [Link]
14. "FPGA AMD Artix™ 7", AMD, Accedido Mayo 16, 2025. [Link]
15. P. Chu, *FPGA Prototyping by VHDL Examples: Xilinx MicroBlaze MCS SoC,* 2ed, Wiley, 2017.