DISEÑO Y CONSTRUCCIÓN DE UN ACCESO DIRECTO A MEMORIA UTILIZANDO EL LENGUAJE VHDL Y LOS DISPOSITIVOS FPGA: UNA REVISIÓN SITEMÁTICA DE LITERATURA

DESIGNING AND BUILDING A DIRECT MEMORY ACCESS USING THE VHDL LANGUAGE AND FPGA DEVICES: A SYSTEMATIC LITERATURE REVIEW

https://doi.org/10.5281/zenodo.4161776

AUTORES: Oscar Cumbicus Pineda [1]*

Dayanna Alvarado Castillo[2]

Josué Ortega Jaramillo[3]

DIRECCIÓN PARA CORRESPONDENCIA: oscar.cumbicus@unl.edu.ec

Fecha de recepción: 10 / 04 / 2020

Fecha de aceptación: 05 / 09 / 2020

RESUMEN

El presente trabajo es una revisión sistemática de la literatura es el resultado de un trabajo previo para la construcción de un Acceso Directo a Memoria (DMA) utilizando el lenguaje de descripción de hardware VHDL, los dispositivos Field-programmable gate array (FPGA) y algunos algoritmos para la programación de estos dispositivos, el objetivo principal de este estudio es proporcionar una metodología eficaz para modelar el controlador de la DMA asegurando un acceso adecuado a los datos optimizando los recursos para cada una de las transacciones relevantes, otro objetivo es conocer las

[1]*Master en Ingeniería Computacional y Sistemas Inteligentes, Universidad Nacional de Loja, oscar.cumbicus@unl.edu.ec

[2]Estudiante de la Carrera de Ingeniería en Sistemas / Computación, Universidad Nacional de Loja, dayanna.alvarado@unl.edu.ec

[3]Estudiante de la Carrera de Ingeniería en Sistemas / Computación, Universidad Nacional de Loja, josue.ortega@unl.edu.ec

diferentes arquitecturas que existen y la configuración de los dispositivos para una implementación mucho más simple y óptima, la investigación literaria incluye un marco de referencia del proceso de Revisión Sistemática de la Literatura sobre estudios primarios centrados en la búsqueda de artículos relacionados con la arquitectura, el diseño y los algoritmos utilizados para construir el Controlador DMA utilizando FPGA y VHDL. Los resultados de la revisión muestran que hay una gran variedad de arquitecturas DMA, el uso de las mismas depende del tipo de transmisión que se quiera realizar y de los tipos de datos involucrados en la transacción, también hay varios modelos de diseño en múltiples lenguajes de programación y modelado, de acuerdo con la arquitectura de la DMA, existe la arquitectura mejorada del controlador que ayuda en gran medida a reducir la latencia de procesamiento, así como la presencia de una arquitectura específica necesaria para la lectura/escritura de imagen y vídeo.

Palabras clave: FPGA, HDL, Arquitectura de computadoras.

ABSTRACT

The present work is a systematic review of the literature is the result of previous work for the construction of a Direct Memory Access (DMA) using the VHDL hardware description language, the Field-programmable gate array devices (FPGA) and some algorithms for the programming of these devices, the main objective of this study is to provide an effective methodology for modeling the DMA controller ensuring proper access to data by optimizing resources for each of the relevant transactions, Another objective is to know the different architectures that exist and the configuration of the devices for a much simpler and optimal implementation, the literature research includes a framework of the Systematic Literature Review process on primary studies focused on the search of articles related to architecture, design and algorithms used to build the DMA Controller using FPGA and VHDL. The results of the review show that there is a great variety of DMA architectures, the use of these architectures depends on the type of transmission you want to make and the types of data involved in the transaction, there are also several design models in multiple programming and modeling languages, according to the DMA architecture, there is the improved architecture of the controller that greatly helps to reduce processing latency, as well as the presence of a specific architecture needed for read/write image and video.

Keywords: FPGA, HDL, Computer architecture.

INTRODUCCIÓN

El acceso directo a la memoria DMA permite a ciertos tipos de componentes de ordenador acceder a la memoria del sistema para leer o escribir independientemente de la Unidad Central de Proceso (CPU).

Muchos sistemas de hardware utilizan DMA, incluyendo controladores de unidades de disco, tarjetas gráficas, tarjetas de red, tarjetas de sonido.

El DMA es esencial en los sistemas integrados, se considera una característica indispensable en todos los ordenadores modernos, permite que dispositivos de diferente velocidad se comuniquen sin someter al CPU a una carga masiva de interrupciones.

Los equipos con canales de acceso directo a la memoria DMA pueden transferir datos desde y hacia dispositivos con menor utilización del CPU que los equipos sin canales DMA.

Una transferencia DMA consiste en copiar un bloque de memoria de un dispositivo a otro, esa transferencia es realizada por el controlador DMA, en lugar de hacerlo con el CPU. Con DMA, el CPU puede iniciar la transferencia, luego realizar otras operaciones mientras la transferencia está en curso y luego recibir una interrupción del controlador DMA una vez completada la transferencia.

El controlador de acceso directo a la memoria (DMAC) suele ser un chipset de la placa base y se considera un dispositivo capaz de controlar la transferencia de datos entre un periférico y la memoria sin intervención del CPU, actúa como bus maestro durante las transferencias DMA y es capaz de solicitar el uso del bus a través de las señales necesarias y la lógica de arbitraje, especificando la dirección de memoria en la que se realiza la transferencia, generando las señales de control del bus que pueden ser señales de tipo operación (lectura/escritura) o señales de sincronización de transferencia(Vanita, 2004).

METODOLOGÍA

Tomando como guía la metodología de revisión sistemática descrita por la autora Barbara Kitchenham(Kitchenham, 2004), se siguió un esquema específico para la revisión, extracción y selección de la información, el cual se describe a continuación:

A. Pregunta de investigación.

B. Palabras clave.

C. Método de revisión.

1. Fuentes y estrategias de búsqueda.
2. Cadenas de búsqueda
3. Criterios de selección de estudios.
4. Extracción de información.

D. Estudios incluidos y excluidos

Además, la herramienta Mendeley se utilizó para gestionar la bibliografía, facilitando el almacenamiento y la buena organización de los artículos en estudio y sus referencias.

A. Pregunta de investigación.

El presente trabajo se centra en la búsqueda de artículos relacionados con la arquitectura, máquinas de estado y algoritmos utilizados para diseñar el controlador DMA utilizando FPGA y VHDL, la pregunta planteada para el desarrollo de este documento es:

¿Qué estudios primarios existen para el diseño y construcción de un Acceso Directo a Memoria utilizando tecnología FPGA y VHDL?

B. Palabras claves.

Después de revisar la literatura y hacer un análisis general de ciertos artículos relacionados con el tema en cuestión, se identificaron las palabras clave presentes en los títulos, resúmenes y conclusiones.

Las palabras clave utilizadas fueron: DMA Controller, DMA State Machine, DMA Architecture, DMA Algorithms, Design DMA on VHDL, DMAC using FPGA.Una vez obtenidas las palabras clave podemos construir la cadena de búsqueda.

C. Método de revisión

1. Fuentes y estrategias de búsqueda.

IEEEXPLORE Library: http://ieeexplore.ieee.org/ (“IEEE Xplore,” n.d.)

GOOGLE SCHOLAR: https://scholar.google.com.ec/ (“Google Académico,” n.d.)

MICROSOFT ACADEMIC: https://academic.microsoft.com/home (“Home | Microsoft Academic,” n.d.)

SCIENCEDIRECT: https://www.sciencedirect.com/ (“ScienceDirect.com | Science, health and medical journals, full text articles and books.,” n.d.)

REFSEEK: https://www.refseek.com/ ("RefSeek - Academic Search Engine," n.d.)

SCOPUS https://www.scopus.com/home.uri ("Scopus preview - Scopus - Welcome to Scopus," n.d.)

2. Cadenas de búsqueda

Teniendo en cuenta la pregunta de investigación y las palabras clave se generaron las cadenas de búsqueda utilizando los operadores lógicos "OR", "SAME" y "AND", obteniendo: *((DMA controller state diagram OR DMA architecture OR Direct Memory Access Controller OR DMA) AND (FPGA OR VHDL OR HDL))*

DIGITAL LIBRARY IEEEXPLORE: ALL ((DMA controller state diagram OR DMA architecture OR Direct Memory Access Controller OR DMA) AND (FPGA OR VHDL OR HDL)) ALL (DMAC MACHINE STATE OR DMAC ARCHITECTURE)

GOOGLE SCHOLAR: ((DMA controller state diagram OR DMA architecture OR Direct Memory Access Controller OR DMA) AND (FPGA OR VHDL OR HDL)) ALL (DMAC MACHINE STATE OR DMAC ARCHITECTURE)

MICROSOFT ACADEMIC: ((DMA architecture OR Direct Memory Access Controller OR DMA) AND (FPGA OR VHDL OR HDL)) ALL (DMAC MACHINE STATE OR DMAC ARCHITECTURE)

SCIENCEDIRECT: ((DMA architecture OR Direct Memory Access Controller OR DMA SAME DMAC algorithms in C)) ALL (DMAC MACHINE STATE OR DMAC ARCHITECTURE)

REFSEEK: ((DMA architecture OR Direct Memory Access Controller OR DMA) AND (FPGA OR VHDL OR HDL))

SCOPUS: ((DMA architecture OR Direct Memory Access Controller OR DMA) AND (FPGA OR VHDL OR HDL))

3. Criterios de selección de estudios.

Para la selección de los artículos de estudio consideramos lo siguiente:

- Sólo artículos con referencias bibliográficas científicas y artículos de revistas.
- Se consideraron artículos desde el año 2013 hasta el año 2019.
- Se verificó que los resultados de la búsqueda sean sólo en el área de electrónica e informática.

- La búsqueda de los artículos para la selección se realiza en idioma inglés.
- Estén enfocados al diseño del controlador DMA en lenguajes de programación como VHDL o HDL.
- Que se utilice FPGA.
- Todos los artículos en estudio deben responder a la pregunta de la investigación.

Los artículos que no se han incluido en el estudio presentan las siguientes características:

- Son artículos informales que excluyen en su proceso de desarrollo metodología científica.
- Son artículos desarrollados y publicados antes del año 2013.
- No presentan bibliografía.
- Todos aquellos que no cumplan con los términos de inclusión.

Obtenidos los resultados de las búsquedas realizadas en cada una de las bibliotecas es necesario seguir un criterio para seleccionar los artículos como relevantes, para ello deben tener las siguientes características:

Presentar en el resumen, información actual del controlador DMA o detallar los algoritmos utilizados para diseñar en VHDL o FPGA y Lenguajes de Programación o describir la máquina de estado del controlador.

Contener información pertinente para su revisión en el resumen, introducción y conclusiones.

Que el artículo en estudio proporcione información centrada en el DMA, en términos de su arquitectura, máquina de estado o diseño.

Se propongan algoritmos para el diseño en VHDL, FGPA y Lenguajes de Programación.

Presentar resultados y conclusiones pertinentes y eficaces.

4. Extracción de Información

Para extraer información de los artículos seleccionados, se verificó que cumplan con los siguientes criterios de extracción.

D. Estudios incluidos y excluidos.

El criterio utilizado para la inclusión de los artículos fue que estos proporcionen información relevante sobre DMA.

Las búsquedas realizadas generaron 50 artículos, de los cuales se registraron 19 coincidencias, es decir, el número de 31 artículos revisados, de los cuales 26 fueron artículos seleccionados según el criterio ya establecidos conforme se lo puede observar en la Tabla 1 que se presenta a continuación:

Tabla 1. Artículos incluidos y excluidos

Artículos				
Bases de Datos	**Encontrados**	**Coincidencias**	**Revisados**	**Seleccionados**
IEEXplore	**29**	**5**	**24**	**20**
Google Scholar	**5**	**5**	**0**	**0**
Microsoft Academic	**8**	**4**	**4**	**3**
Science Direct	**3**	**2**	**1**	**1**
Refseek:	**3**	**1**	**2**	**2**
Scopus	**2**	**2**	**0**	**0**
Total	**50**	**19**	**31**	**26**

ANÁLISIS DE LOS ESTUDIOS PRIMARIOS SELECCIONADOS

Una vez seleccionados los 26 artículos Tabla 2, se analizaron y se los dividió en 3 campos: Arquitectura del DMA, Diseño del DMA y Algoritmos utilizados en los DMA, a continuación, se presentan los hallazgos más relevantes de los artículos divididos en los 3 campos

Tabla 2. Artículos seleccionados

Bases	Autores	Título del Artículo	Código Asignado
IEEXPLORE	(Shanehsazzadeh & Sadri, 2017)	Area and performance evaluation of central DMA controller in Xilinx embedded FPGA designs	Art01
	(Mantovani, Cota, Pilato, Di Guglielmo, & Carloni, 2016)	Handling large data sets for high-performance embedded applications in heterogeneous systems-on-chip	Art02
	(Biglari, Qasemi, & Pourmohseni, 2013)	Maestro: A high performance AES encryption/decryption system Proceedings	Art03
	(Morales, Duran, & Roa, 2019)	A Low-Area Direct Memory Access Controller Architecture for a RISC-V Based Low-Power Microcontroller	Art04
	(Rota, Caselle, Chilingaryan, Kopmann, & Weber, 2015b)	A PCIe DMA Architecture for Multi-Gigabyte Per Second Data Transmission	Art05
	(Gonzalez, Lopez, Mozos, & Sarmiento, 2015)	FPGA Implementation of the HySime Algorithm for the Determination of the Number of Endmembers in Hyperspectral	Art06
	(Khedkar & Khade, 2017)	High speed FPGA-based data acquisition system	Art07
	(Kavianipour, Muschter, & Bohm, 2014)	High Performance FPGA-Based DMA Interface for PCIe	Art08
	(Nguyen, Dong, & Tran, 2019)	A reconfigurable multi-function DMA controller for high-performance computing systems	Art09

	(Aswal, Singh, & Yadav, 2015)	Designing of DMA controller using VHDL	Art10
	(Ng, Choi, & So, 2013)	Direct virtual memory access from FPGA for high-productivity heterogeneous computing	Art12
	(Didariya, Jasrotia, Gupta, Gurjar, & Tripathi, 2016)	Design and Implementation of Generic DMA using VHDL	Art13
	(W. Li, Zhao, Liu, & Chen, 2018)	SMEFF: A scalable memory extension fabric for FPGA	Art14
	(Ji et al., 2012)	DMA-Assisted, Intranode Communication in GPU Accelerated Systems	Art15
	(Tumeo, Monchiero, Palermo, Ferrandi, & Sciuto, 2008)	Lightweight DMA Management Mechanisms for Multiprocessors on FPGA	Art17
	(Bernardi et al., 2017)	A DMA and CACHE-based stress schema for burn-in of automotive microcontroller	Art20
	(Wang, Wang, Zhou, & Wang, 2014)	Design and implementation of a flexible DMA controller in video codec system	Art22
	(Ye, Li, Du, & Cai, 2017)	Design of data transmission system for speed measurement radar between ARM and FPGA based on embedded Linux	Art23
	(Ammendola et al., 2013)	Virtual-to-physical address translation for an fpga-based interconnect with host and gpuremote dma cap abilities	Art25
	(Enami, Kawakami, & Yamazaki,	DMA-driven control method for low power sensor node	Art26

MICROSOFT ACADEMIC	2015)		
	(Zhao, Li, Zhang, Lin, & Chen, 2017)	Research on FPGA timing optimization methods with large on-chip memory resource utilization in PCIe DMA	Art16
	(Rota, Caselle, Chilingaryan, Kopmann, & Weber, 2015a)	A new DMA PCIe architecture for Gigabyte data transmission.	Art19
	(Y. Li, Cai, & Xu, 2018)	Improved DMA Algorithm for the PXIE Bus	Art21
SCIENCE DIRECT	(Fjeldtvedt & Orlandić, 2019)	CubeDMA – optimizing three-dimensional DMA transfers for hyperspectral imaging application	Art24
REFSEEK	(Tiwari & Advanced, 2011)	AMBA dedicated DMA controller with multiple masters using VHDL	Art11
	(Mukherjee et al., 2016)	An efficient approach to evaluate PCIe DMA design and DMA performance for Common Readout Unit (CRU).	Art18

Estudios sobre Arquitectura del DMA

El artículo Art01 muestra un sistema basado en PowerPC y un controlador de memoria multipuerto (MPMC) donde existe el módulo de conexión FPGA a DDR SDRAM. El módulo PLB utiliza dos interfaces, una de las cuales es el Master, que podrá iniciar transacciones a través del bus PLB y el esclavo de respuesta única para solicitar el bus PLB DMA, la central tiene dos esclavos y un maestro conectados. A través del interfaz esclavo, el usuario puede configurar el controlador DMA central con la ayuda del CPU. La interfaz maestra se utiliza para realizar operaciones DMA. El controlador DMA central se puede utilizar para realizar diferentes modelos de operación de transferencia de datos en el caso 1 donde hay un módulo personalizado y la transferencia es directa desde el módulo a la memoria y en el caso 2 donde hay dos módulos personalizados donde la transferencia

puede ser primero entre estos módulos y luego a la memoria. Y finalmente, el controlador DMA central puede ser utilizado para transferir datos entre dos nodos con alta eficiencia y sin imponer cargas sustanciales en el núcleo de la CPU (Shanehsazzadeh & Sadri, 2017).

El artículo Art02 presenta una arquitectura robusta especializada en DMA y Linux que demuestra un gran rendimiento en diferentes escenarios de trabajo y está basada en prototipos FPGA. Esta arquitectura utiliza múltiples aceleradores para una mejor captura de datos y reducir la latencia en la memoria externa, también este tipo de arquitectura se utiliza para el procesamiento de grandes conjuntos de datos, ya que combinan hardware y software al mismo tiempo. El diseño de esta arquitectura ha sido evaluado a través de un sistema completo basado en FPGA, en el que se demuestran cuatro aspectos principales: viabilidad, baja sensibilidad a las características de carga, baja sensibilidad a la colocación de aceleradores y escalabilidad de los conjuntos de datos (Mantovani et al., 2016).

En el artículo Art03 se explica que los controladores DMA se utilizan para controlar la transmisión de datos entre el motor y las memorias AES. Además, explica un trabajo donde se implementa un acceso directo a la memoria, este acceso a memoria presenta un alto rendimiento, sin embargo, se presenta un costo eficiente en el sistema de cifrado/descifrado AES. El sistema completo funciona a 100 MHz y el motor AES es capaz de cifrar/descifrar un bloque en cada ciclo de reloj, lo que resulta en un rendimiento de núcleo de 12,8 Gbps. Sin embargo, el ancho de banda de la memoria limita el rendimiento del sistema a 1,85 Gbps, que sigue siendo un millón de veces más rápido que el software de aplicación pura del código de referencia en el procesador NIOS de la misma plataforma. Una tarjeta FPGA con mejor ancho de banda de memoria podría mejorar los resultados (Biglari et al., 2013).

El artículo Art04, muestra un esquema de la arquitectura principal del controlador DMA donde cada periférico puede realizar transacciones de lectura y escritura utilizando uno o más canales para acceder a la memoria. Un único FIFO almacena todas las transacciones concedidas y almacena un ID para identificar el canal a reconocer, el tipo de transacción y los datos de escritura para las operaciones de escritura. La función de configuración de canales DMA permite el diseño para aplicaciones de baja potencia. Los recursos compartidos y el ancho de palabra reducido permiten un bajo consumo de energía (Morales et al., 2019).

Art05, este documento presenta la arquitectura del motor DMA, la arquitectura DMA utiliza dos motores diferentes para transmitir y recibir datos (TX y RX) con el núcleo PCIe. El motor RX se implementa utilizando la misma arquitectura que el motor TX. La arquitectura también incluye la tabla de direcciones y una base de registro de espacio de direcciones (BAR). La interfaz entre los motores TX/RX y la lógica de usuario es una interfaz tipo FIFO con un ancho de 128 bits, de acuerdo con el ancho de los datos de entrada/salida del núcleo PCIe, y que funciona a 250 MHz. Esta arquitectura muestra un rendimiento del 94% del límite teórico y proporciona una interfaz sencilla similar a la FIFO para la lógica de usuario (Rota et al., 2015b).

El Art08 manifiesta que los sistemas de adquisición de datos a menudo utilizan PCIe para comunicarse entre las tarjetas de procesamiento basadas en FPGA y el ordenador central y requieren la transferencia de grandes cantidades de datos a través de este canal. Pero los servicios limitados que ofrece la FPGA en modo esclavo autónomo suelen ser un cuello de botella, lo que obliga a muchos a utilizar sistemas integrados en la FPGA, con consecuencias negativas para el rendimiento y la superficie disponible en la FPGA. Sin embargo, un diseño autónomo podría gestionar la transferencia con un mayor rendimiento y una huella más pequeña en la FPGA si utiliza DMA y especialmente DMA de bus maestro. La tasa de transferencia más alta alcanzada para una sola transferencia de búfer fue de 784 MBytes/s leídos y 748 MBytes/s escritos para la tarjeta de evaluación de Virtex-7. El límite teórico de un bus 2PCIe generando 4 carriles está de acuerdo con los cálculos de aproximadamente 1500MB/s. Esto se debe a la sobrecarga de codificación y señalización en la capa de enlace físico y al control del flujo de tráfico de la capa de enlace de datos. Por lo tanto, tendríamos una eficiencia del 52% con los tamaños de paquetes actuales (8KB) (Kavianipour et al., 2014).

El artículo Art09 describe un ReDMAC el cual está diseñado para mantener el papel como un adaptador entre los sistemas de procesamiento basados en ARM AMBA con aceleradores de hardware. ReDMAC utiliza una interfaz y conectividad de sistema en chip. Incluye una interfaz maestra AHB para acceder a la memoria del sistema y una interfaz esclava AHB para recibir comandos DMA desde la CPU. Además, ReDMAC también tiene otra interfaz para la compresión de CPU o periféricos que solicitan una sesión de DMA.

Desde la perspectiva de la estructura, ReDMAC incluye dos partes: Envoltura del controlador DMA y núcleo del controlador DMA. La envoltura debe hacer que la interfaz del núcleo del controlador DMA sea compatible con el bus AHB y la interfaz del acelerador, permitiendo así que el núcleo del controlador DMA transfiera datos entre la memoria y el acelerador. El kernel del controlador del DMA consiste en los tres bloques principales que son Registry File Control, Context Generator Configuration (CCG), y la Unidad de Control (CU). Por encima de todo, para proporcionar la capacidad de reconfigurar en tiempo real (Nguyen et al., 2019).

En el artículo Art10 se describe un diseño de controlador DMA para AHB del bus AMBA con varios masters. Un bus maestro es capaz de iniciar la operación de lectura y escritura, proporcionando información de dirección y control. El controlador DMA, que consiste en una memoria y una unidad de control, es el módulo básico de este proyecto. La unidad de control controlará la operación de lectura y escritura. Cuando el periférico 1 desea leer datos de la memoria o escribir datos en la memoria, el periférico 1 enviará una señal de activación alta. Cuando el periférico 1 quiere leer datos de la memoria, la señal de la barra de lectura/escritura será alta. Del mismo modo, hará que la lectura de la barra de escritura sea baja para escribir los datos en la memoria. El mismo proceso se lleva a cabo con el periférico 2; pero al mismo tiempo, sólo un periférico puede funcionar, es decir, sólo un periférico puede acceder desde el BUS para leer o escribir los datos. Así que también diseñan un árbitro para dar prioridad entre el periférico 1 y el periférico 2. El árbitro tomará la petición del periférico 1 y del periférico 2 y dará la respuesta al periférico 1 y al periférico 2 respectivamente. Pero cuando ambos periféricos dan la petición en un momento dado, el árbitro responderá al periférico 1 y al periférico 2 alternativamente. Un árbitro más se usa para dar prioridades entre el sistema DMA y el anfitrión (Aswal et al., 2015).

El artículo Art11 nos propone un controlador DMA 8237A que está configurado para realizar transferencias por DMA, teniendo la CPU para programar sus 16 registros internos a través de un conjunto de 16 puertos de E/S. La programación se realiza a través de las líneas de control, y (selección de chip), y las líneas de datos DB0-DB7. Cuando se realiza una transferencia DMA real, el controlador DMA 8237A toma las líneas de control y

dirección de la CPU y genera estas señales por sí mismo. Esto se debe a que están presentes las líneas de CLOCK y READY, las líneas de control y las líneas de dirección. Dentro de cada controlador DMA 8237A hay cuatro canales DMA; cada canal puede programarse independientemente para realizar su propia transferencia DMA, es decir, un controlador DMA 8237A se comporta como cuatro controladores DMA coordinados de tal manera que en un momento dado sólo uno de ellos toma el control del bus (Tiwari & Advanced, 2011).

El artículo Art12 describe una arquitectura donde la FPGA permite al gateware acceder directamente al espacio de memoria virtual como parte del proceso de ejecución sin involucrar a la CPU. Para esta Arquitectura de controlador DMA se implementa un búfer de direcciones en caché junto con el gateware FPGA para proporcionar un mejor tiempo de ejecución, además de mapeo entre las direcciones de memoria física y virtual. El subsistema de memoria es una parte indispensable de los sistemas de procesadores modernos donde se almacenan tanto las instrucciones del programa como los datos durante el tiempo de ejecución (Ng et al., 2013).

En el artículo Art13 se ha diseñado un controlador genérico de acceso directo a la memoria (DMA) que es reconfigurable dinámicamente y tiene una característica de control diferente. Este controlador DMA funciona en dos modos diferentes. En el primer modo, hay un bus de datos fijo (8 bits), un bus de direcciones (16 bits) y un registro de recuento de palabras (16 bits) y en el segundo modo hay un bus de datos variable (8, 16, 32, 64 bits), un bus de direcciones (16,24,32 bits) y un registro de recuento de palabras (8,16,24,32 bits). El controlador DMA tiene 4 pares de registros de canal. Cada uno está provisto de un registro de recuento de terminales de 16 bits y un registro de direcciones de 16 bits. En un sistema informático, la DMA es una característica mediante la cual los dispositivos de entrada/salida pueden acceder a la RAM del ordenador independientemente de la CPU. Sin DMA, la CPU no puede realizar otras tareas al mismo tiempo que se transfieren los datos. Pero con DMA, es posible que la CPU opere en otras tareas durante la transferencia de datos (Didariya et al., 2016).

En términos de capacidad de memoria y ancho de banda, en el artículo Art14 se introduce un nuevo diseño de organización del modelo de DMA SMEFF. Donde el bus serie de alta velocidad basado en FPGA se aprovecha para construir una red de memoria, que

proporciona x5 en ganancias de capacidad de memoria y hasta x3.6 en mejoras de ancho de banda de memoria en comparación con los sistemas de memoria FPGA actuales. También es superior a los sistemas de memoria extendida basados en PCIe. SMEFF introduce el concepto de un sistema de memoria asíncrona escalable y multinivel. En la latencia de la memoria, el consumo de energía y los datos, la tecnología DMA de M-TO-M alcanza una latencia de hasta x3, y una reducción de potencia del 21,1% hasta el 61,1% en comparación con la última generación de memoria basada en FPGA (W. Li et al., 2018).

En el artículo Art15 se ha implementado técnicas eficientes entre aceleradores dentro del mismo nodo asistidas por DMA, permitiendo al programador suministrar los búferes de los dispositivos directamente llamadas de la MPI(Mecanismos para el intercambio directo entre pares asistido por DMA), que involucran al host y a la GPU, los datos entre pares asistidos por la DMA ofrece mayores beneficios cuando se aplica a los dispositivos de la GPU que están cerca, la transferencia de datos asistida por DMA es capaz de aprovechar el rápido movimiento de los datos dentro del dispositivo GPU, lo que resulta en una mejora del ancho de banda y la latencia en orden de magnitud en comparación con shm (memoria compartida del lado del host) que nos permite obtener el mejor rendimiento (Ji et al., 2012).

En el documento Art16 se realizó la transmisión de acceso directo a la memoria (DMA) en modo PCIe Gen2 basado en Kintex-7 FPGA, y se resolvieron principalmente los problemas de sincronización causados por el buffer de flujo de datos en el chip de gran capacidad.Se propone un mecanismo de espera de transferencia para hacer la optimización del tiempo, el uso del método de espera de transferencia puede resolver errores de sincronización. Mientras tanto, el mecanismo de espera hace que el ancho de banda de transferencia disminuya (Zhao et al., 2017).

El artículo Art17 presenta un sistema multiprocesador FPGA que adopta mecanismos de Acceso Directo a la Memoria (DMA) para mover datos entre la memoria externa y la memoria local de cada procesador. La interfaz permite programar la arquitectura multiprocesador integrada en FPGA con DMAs simples, que utilizan las mismas técnicas DMA adoptadas en los multiprocesadores de alto rendimiento, permitiendo una mejora del 57% sobre el tiempo de ejecución de un conjunto seleccionado de puntos de referencia. Por otro lado, la ventaja de utilizar una unidad DMA es que, en lugar de ejecutar varias

instrucciones de montaje para realizar la copia de memoria, se realiza un simple comando a la unidad, haciendo que el procesador espere a que se complete la transferencia. Sin embargo, esto significa que mientras se realiza la transferencia, el procesador está desperdiciando ciclos útiles esperando a que la transferencia termine. Por lo tanto, se aplica la técnica comúnmente conocida como doble búfer, ya que utiliza búferes idénticos; A y B, para realizar el cálculo en uno de ellos mientras que los datos son transferidos por el otro. El almacenamiento temporal de datos puede ampliarse utilizando tantos búferes como espacio disponible en la memoria local (Tumeo et al., 2008).

En el artículo Art18 el controlador DMA propuesto con RTL Verilog sintético fue implementado e integrado en el acelerador M2G, utilizado para aplicaciones, incluyendo el cálculo FFT (Fast Fourier Transform) y la multiplicación general de matrices (GEMM), para evaluar las optimizaciones. Para el cálculo FFT, el rendimiento del controlador DMA tradicional está limitado por su baja utilización de ancho de banda, sólo puede transponer un elemento de matriz por ciclo de reloj. Por ello, su rendimiento es la mitad del diseño optimizado. El controlador DMA optimizado logra un ancho de banda de movimiento de datos similar al máximo teórico. En promedio, su rendimiento es sólo un 2,3% inferior al del diseño ideal. Para aplicaciones reales, oculta eficientemente la latencia del movimiento de los datos detrás de los cálculos y trabaja estrechamente con un diseño ideal (Mukherjee et al., 2016).

La arquitectura propuesta en el artículo Art19 adopta una nueva estrategia para evitar la reducción de velocidad: todos los descriptores necesarios se almacenan en la memoria FPGA y el motor DMA carga automáticamente la dirección física del bloque de memoria durante la operación de escritura. El motor DMA se está ampliando actualmente para soportar arquitecturas PCIe Gen3 y multi-motor master-slave; el controlador utiliza un buffer anular en la memoria del usuario con un tamaño limitado (8 Mb), lo que también reduce considerablemente la cantidad de memoria necesaria para almacenar los descriptores dentro de la FPGA. Cuando todos los datos han pasado de FPGA a PC, el DMA actualiza el puntero WR con el último descriptor e informa al controlador sobre la cantidad exacta de datos escritos en la última página 4k. También se genera una interrupción para informar a la CPU sobre el final de la transmisión. Gracias a la gestión de errores PCIe y handshake, no

se han detectado errores en los tamaños de datos transferidos de varios cientos de GBs (Rota et al., 2015a).

El artículo Art20 presenta esquema de estrés basado en DMA y CACHE para la quema del microcontrolador automotriz mediante el cual se presenta una metodología para optimizar los procedimientos de estrés durante la fase Burn-In. Muestra e ilustra un nuevo enfoque para realizar esfuerzos durante el flujo de BI basado en transferencias DMA y memoria caché. Se presentan varios escenarios. El enfoque propuesto consiste principalmente en reproducir con el DMA disponible el acceso de prueba de memoria realizado por un motor BIST. Este esquema garantiza una paralelización completa aprovechando las memorias caché y proporciona la ventaja de calentar la superficie del chip a alta temperatura, implementando así un mejor estrés térmico (Bernardi et al., 2017).

En el artículo Art21 se propone un nuevo diseño de DMA de bus PXIE que se basa en un método de programación DMA empalmado dinámicamente, combinando áreas de memoria adyacentes, la forma en que se accede a la petición reduce el número de interacciones PC-a-hardware e interrumpe las peticiones. La plataforma manejada para la prueba se basa en el dispositivo Xilinx Virtex-7 serie xc7v585tffg1761-1, desarrollado en VerilogHDL, que analiza el rendimiento de la transmisión en modo DMA, que se transmite a un paquete de datos de tamaño fijo, y el tiempo que tarda el DMA en leer y escribir el paquete de datos de tamaño fijo se obtiene en el controlador, y finalmente se calcula la velocidad de transmisión de lectura/escritura de DMA (Y. Li et al., 2018).

En el artículo Art22 de acuerdo con la demanda del sistema y las características de los datos del códec de vídeo, se realiza un controlador DMA de diferentes canales que permite el Algoritmo de Conversión de Dirección de Bloque, el Modelo de Descriptor de canal, la Técnica de Precorteo, Este controlador DMA personalizado fue diseñado en lenguaje HDL de Verilog y fue sintetizado con éxito en el circuito de nivel de puerta con el proceso CMOS SMIC de 65 nm por Synopsys Design Compiler, los usos del algoritmo de conversión de direcciones de bloques, el modelo de descriptor de canal, el árbitro inteligente, la técnica de pre-captura y los búferes FIFO hacen que el controlador DMA sea más flexible y eficiente para el procesamiento de imágenes y vídeo. Después de la verificación funcional del sistema en la plataforma del acelerador de hardware, el diseño

propuesto se implementa utilizando el proceso CMOS SMIC de 65nm y alcanza la frecuencia de 625MHz con sólo 23.6K puertas lógicas. Los resultados experimentales muestran que la eficiencia de la transferencia recibe una mejora de alrededor de 2 ~ 4 veces (Wang et al., 2014).

El articulo Art23 describe un ARM, basado en DMA para la ejecución de transacciones, que aumenta la velocidad de transmisión, el diseño genera un FIFO asíncrono de 32 bits de ancho de datos y profundidad de 2048 palabras en la FPGA, la escritura es el reloj del sistema FPGA que es de 40MHz, el chip i.mx6 permite la frecuencia de la salida de datos de la FPGA para que haya una transmisión correcta, para evitar la pérdida de datos y obtener datos de transmisión correctos. El reloj EIM externo de i.mx6 es de 132 MHz, lo que permite una transmisión de datos a alta velocidad. Después de la finalización de la FPGA y la generación de una señal válida, comprueba si el FIFO está lleno (2048 bits), si es cierto el FIFO se detiene, los datos se escriben y al mismo tiempo se activa la señal de escritura (Ye et al., 2017).

En [24] Art24, se manifiesta que una de las formas más eficientes de transmitir datos desde la memoria es estableciendo comunicación directa entre la memoria y el núcleo de procesamiento en la FPGA y excluyendo la Unidad Central de Procesamiento (CPU) de este camino crítico, un controlador convencional de Acceso Directo a la Memoria (DMA) realiza una transferencia de ubicaciones de memoria contiguas que sólo soportan patrones de acceso a datos simples. AXI DMA puede realizar transferencias de datos BIP y órdenes de bloque BIP, pero surgen desafíos para las transferencias BSQ. La eficiencia de AXI DMA en modo 2D está limitada por el requisito de alineación de direcciones de 64 bits, mientras que Video DMA no es lo suficientemente flexible como para ser utilizado para imágenes HSI. Por estas razones, un núcleo DMA personalizado, CubeDMA, especializado en aplicaciones hiperespectrales, puede ordenar los píxeles secuencialmente o por bloques (Fjeldtvedt & Orlandić, 2019).

El artículo Art25 hace mención de un acceso directo a la memoria de modo remoto en el cual ya sea desde el host o desde la memoria de la GPU, en la fragmentación el flujo de datos en paquetes se envía a los puertos de destino correspondientes, dependiendo de la operación solicitada. En el lado de la recepción, proporciona soporte de hardware para el

Protocolo de acceso remoto directo a la memoria (RDMA), que permite transferencia remota de datos a través de la red sin la participación de la CPU del nodo remoto. APEnet+ utiliza el Remote Direct Protocolo de acceso a la memoria; con RDMA, donde la NIC realiza las transferencias lectura/escritura mediante la memoria, sin intervención del sistema operativo ni de funciones (Ammendola et al., 2013).

Artículo Art26.- La DMA generalmente transfiere datos a la mayoría de los registros periféricos, como el registro de estado y el registro de comandos, también utiliza la función de dispersión para transmitir una serie de datos de varias fuentes a varios destinos; por otro lado, el control de rutina de la MCU se considera como una transferencia secuencial de datos entre memorias y registros, para ello debe proceder a realizar 3 procesos para la sustitución por el control controlado por la CPU, el primero se realiza emitiendo el comando de arranque, el segundo confirma si el buffer de transmisión está vacío y finalmente envía una dirección a través de la opción de buffer de transmisión y luego es sustituido por las dos transmisiones. Las instrucciones de la DMA, los datos se transfieren como 1 START y la dirección se calcula preliminarmente y se almacena en memoria de acceso aleatorio (RAM), cuando se opera el procesamiento rutinario iterativo, los resultados experimentales revelaron que el control controlado por DMA redujo el consumo de energía al 37% al agregar la arquitectura de hardware, considerando que las MCUs existentes no pueden cambiar los modos de operación sin las CPUs, sin embargo, la arquitectura que proponemos permite a las DMA cambiar los modos de operación (Enami et al., 2015).

Estudios sobre el Diseño de un DMA

En Art01, la lectura/escritura de búfer DMA individual se realiza como bloque de transferencia; durante una operación de escritura, la aplicación de usuario coloca los datos en un búfer y envía una solicitud de escritura estándar al controlador, que a su vez almacena los datos en una memoria contigua. A partir de ese momento, el controlador inicia el DMA y se pone a dormir. Cuando se ejecuta el DMA, se envía una interrupción al controlador, donde se activará la subrutina de interrupción The Sleeping Process. A continuación, se verifica el resultado de la transferencia y se devuelve a la aplicación de usuario. El script vectorizado, que se coloca en una biblioteca de funciones y opera en el

espacio de usuario, fusiona los búferes en uno solo y utiliza el script ordinario para transferir los datos. Esto es de la aplicación principal, cada búfer es de sólo 12B de tamaño. Por eso no usamos directIO. Sin embargo, utilizando la interfaz asíncrona de la biblioteca de Iibaio, podemos iniciar múltiples operaciones de escritura a la vez y evitar tener un interruptor de contexto para cada operación (Shanehsazzadeh & Sadri, 2017).

Los artículos Art09, Art10, Art1, describen que VHDL es un lenguaje de propósito bastante general, puede leer y escribir archivos en el host del ordenador. La ventaja clave de VHDL cuando se utiliza para el diseño del controlador DMA es que permite describir (modelar) y verificar (simular) el comportamiento requerido del sistema antes de que las herramientas de síntesis traduzcan el diseño en hardware real (puertas y cables) (Nguyen et al., 2019), (Aswal et al., 2015), (Tiwari & Advanced, 2011).

En Art11, se describe una simulación que nos indica que después de la aplicación de la señal de reloj, el periférico 1 y el periférico 2, ambos solicitan el bus para la operación de escritura a la unidad de control. La unidad de control permitirá el acceso al bus de forma alternada a ambos periféricos. Después de acceder al bus, el periférico envía una dirección de 6 bits, datos de 8 bits y una señal de lectura/escritura a la unidad de control. Los datos proporcionados por el periférico se escriben en la posición de memoria, cuya dirección también es proporcionada por el periférico. Los datos proporcionados por el periférico 1 y el periférico 2 se escriben en la posición de memoria, cuya dirección también es dada por el periférico 1 y el periférico 2 respectivamente. Los datos de la posición de memoria seleccionada son leídos por el periférico que aparece en la línea de salida de datos (Tiwari & Advanced, 2011).

Estudios sobre Algoritmos para diseñar un DMA

Este artículo Art07 presenta el primer diseño FPGA para el uso e implementación del algoritmo HySime, junto con el algoritmo I/O Communications. El algoritmo comienza estimando las matrices de correlación de señal y ruido y luego selecciona secuencialmente el subconjunto de vectores apropiados que mejor representan el subespacio de la señal en la dirección del error cuadrado medio mínimo (MSE). El sistema incluye un módulo de acceso directo a la memoria (DMA) y se implementa una técnica de preconfiguración para ocultar la latencia de las comunicaciones de entrada/salida. La versión de hardware del

algoritmo HySime puede superar significativamente (en términos de tiempo de cálculo) al software, lo que hace que este sistema reconfigurable sea atractivo para procesar los datos hiperespectrales con los que se está trabajando. Además, la existencia de FPGAs ofrece la atractiva posibilidad de seleccionar de forma adaptativa un algoritmo de procesamiento hiperespectral (Khedkar & Khade, 2017).

En el artículo Art08 hay dos algoritmos. El primero muestra la operación con un trigger externo cuando se produce un flanco positivo en la señal Data_valid. La operación comienza cuando la señal de entrada analógica se convierte en digital mediante el uso del convertidor analógico-digital (ADC), la salida ADC es de 2 bits que se convierten 16 bits por serialización. La salida digitalizada de 16 bits se almacena en la memoria del chip la misma que al llenarse los datos se almacenan en otra memoria y al llenarse la segunda memoria se genera una interrupción, la misma que sirve para activar el DMA. El DMA está programado para hacer transferencias en modo rafaga, este tipo de transferencia ocurre continuamente, esto muestra la implementación de un buffer circular/memoria en DDR3. La transferencia de datos ocurre continuamente hasta que Data_valid es positivo y el segundo muestra el aumento de la operación de un trigger externo después de que se produce un flanco positivo en la señal data_valid, lo que implica que Data es válido sólo cuando la señal Data_valid es alta.Se produce el flanco de subida de la señal de transferencia de datos y con el flanco de bajada se detiene cuando se detecta el flanco de subida de la señal Data_valid, la transferencia de datos se realiza de la misma manera que el primer algoritmo y se detiene con su flanco de bajada, también utiliza un PRE-TRIP y un POST-TRIP que se aplican antes del punto de evento y los otros después de él respectivamente, para asegurar que no hay pérdida de datos (Kavianipour et al., 2014).

HALLAZGOS

Una vez realizado el estudio de los artículos se pudo determinar que describen varias arquitecturas del controlador DMA. Los artículos 01 y 05 describen una arquitectura desarrollada para la transferencia de grandes cantidades de datos con un rendimiento del 94% del límite teórico que proporciona una interfaz sencilla similar a la FIFO.El Art05 añade contenido procedente de transferencias a gran velocidad de datos entre subsecciones, con lo que se consigue un uso muy reducido de los recursos. Art03 cuenta con un eficiente

sistema de cifrado/descifrado AES. Todo el sistema funciona a 100 MHz y el motor AES es capaz de un bloque de cifrado/descifrado en cada ciclo de reloj. El Art 04 menciona una arquitectura DMA donde cada periférico permite emitir transacciones de lectura y escritura a través de uno o varios canales, estas transacciones se almacenan en un único FIFO donde se almacena un ID para reconocer el canal y el tipo de transacción. El uso de esta arquitectura permite el diseño de aplicaciones de baja potencia. Art08 propone una arquitectura que puede ser utilizada para transferir datos entre dos nodos con alta eficiencia y sin imponer cargas sustanciales en el núcleo de la CPU. El Art.10 describe una arquitectura conocida como ReDMAC que funciona como un adaptador entre sistemas con acelerador de hardware. El Art.11 especifica cómo crear un controlador DMA con varios buses maestros. El Art.10 propone un controlador DMA8237A que está configurado para realizar transferencias DMA. Los artículos 9 y 10 especifican que la mejor manera de crear un controlador DMA se basa en las especificaciones del protocolo AMBA AHB, ya que reduce la complejidad y el costo. Art20, tiene un nuevo enfoque que garantiza una paralelización completa aprovechando las memorias caché y ofrece la ventaja de calentar a alta temperatura la superficie del chip, implementando un mejor estrés térmico. El Art16 describe una optimización de los tiempos de transmisión mediante el diseño de una conexión de interfaz FPGA entre FIFO y el núcleo IP PCIe para realizar la transmisión de secuencia de datos de escritura DMA. El Art19, a diferencia del anterior, adopta una nueva estrategia, donde la lista de descriptores DMA se almacena dentro de la FPGA y no en el sistema de memoria central. Esta arquitectura también utiliza un buffer anular en la memoria del usuario con un tamaño limitado de 8MB, lo que reduce considerablemente la cantidad de memoria necesaria. El Art22 cuenta con un controlador DMA de cuatro canales que puede realizar trabajos de transferencia de datos en el sistema H.264 / RVC. Los usos del algoritmo de conversión de direcciones de bloque, el modelo de descriptor de canal, el árbitro inteligente, la técnica de pre-captura y los buffers FIFO hacen que el controlador DMA sea más flexible y eficiente para el procesamiento de imágenes y vídeo. El Art23 presenta una arquitectura que permite diseñar una interfaz para la comunicación entre ARM y la FPGA que genera un FIFO asíncrono de 32 anchos de datos y 2048 palabras de profundidad, además utiliza una señal de reloj de 132 Mhz que permite una transmisión de

datos a alta velocidad. Art24 presenta un diseño arquitectónico para un dispositivo de interconexión APEnet+ con 6 enlaces bidireccionales que permiten la comunicación con los nodos a través de RDMA para reducir los tiempos de latencia y mejorar el ancho de banda en las aplicaciones informáticas, un buffer TLB que permite la traducción avanzada, permite también el control del manejo de errores e introduce el motor DMA para las posteriores transmisiones de datos. El Art25 presenta una integración de la DMA para reducir el consumo de energía que genera resultados de 150 A/MHz a 51A/MHz, utiliza la función scattergather para transmitir una serie de datos de varias fuentes a varios destinos, envía una emisión de inicio, con la implementación de la DMA se pueden habilitar los nodos de operación que generaron un error en la MCU que no permitió un bajo consumo de energía.

Por otro lado de acuerdo al diseño generalmente se menciona que el tamaño del buffer está bastante regulado y para mejorarlo se debe aumentar el tamaño del buffer dentro de la transferencia, también se puede mejorar cambiando el chipset ya que esto nos proporciona una mejora en el tamaño máximo de la carga útil y esto afecta directamente el tamaño de la ráfaga de datos permitida en el bus, dentro del diseño y adaptabilidad del código se puede diseñar fácilmente un DMA muy flexible, altamente controlable y compatible, la mayoría de las mejoras se dan aumentando los buffers y el propio chipset de la placa madre que serían características de las placas principales de acuerdo a sus series, Como se define en los artículos 9, 10 y 11, el mejor lenguaje para diseñar es el VHDL porque es un lenguaje de uso bastante general, lo que nos da más ventajas sobre otros lenguajes, como escribir un banco de pruebas que verifica la funcionalidad del diseño utilizando archivos en el ordenador central para definir estímulos, interactuar con el usuario y comparar resultados.

Según Algoritmos, una de las formas de transferir datos desde la memoria es estableciendo una comunicación con el procesador, este es el caso del Art24 que habla de una forma de transferir dichos datos utilizando dispositivos SoC que adaptan y combinan arquitecturas informáticas como CPUs, GPUs, FPGAs y DSPs, que se utilizan en la partición del algoritmo, una de las formas más eficientes de transmitir datos desde la memoria es estableciendo una comunicación directa entre la memoria y el núcleo de procesamiento de la FPGA y excluyendo la Unidad Central de Procesamiento (CPUs), siendo un poco

diferente en el caso del Art06, que utiliza un sistema que incluye un módulo de Acceso Directo a la Memoria (DMA) e implementa una técnica de preconfiguración para ocultar la latencia de las comunicaciones de entrada/salida, ya que un ejemplo muestra una versión de hardware del algoritmo HySime que puede superar significativamente (en términos de tiempo de cálculo) al software, lo que hace que este sistema reconfigurable sea atractivo para el procesamiento de los datos hiperespectrales con los que se desea trabajar, pero si sólo se desea trabajar con datos en general, se debe optar por el algoritmo que se describe en el Art24 y que se ocupa directamente de la transferencia de datos conjuntamente con el algoritmo que implementa un trigger externo después de que se produce un flanco positivo en la señal data_valid debido a que en el proceso de operación implementa un valor de datos externos después del punto de evento y algunos datos antes del punto de evento llamados datos POST-TRIP y PRE-TRIP respectivamente, que ayudan a verificar que la señal de entrada es la misma que la señal de salida y que por lo tanto no pierde ningún dato al realizar el proceso completo.

Finalmente, es más eficiente utilizar una arquitectura de controlador DMA mejorado, ya que el DMA tradicional tiene una transmisión de interrupción de un byte cuando se utiliza una transferencia de bytes; si se necesita la siguiente transferencia de bytes, debe seguir interrumpiendo, lo que lleva mucho tiempo, mientras que la función principal del DMA mejorado es eliminar las señales de interrupción innecesarias, aumentando así el ancho de banda y la velocidad de transmisión. La mejor arquitectura presenta una representación de la gestión de tareas de memoria que permite una baja latencia en la transmisión de datos generando resultados desde 1900ns a 124ns con el buffer de predicción de traducción que mejora el ancho de banda e iniciando una transferencia de host a host de 1520 MB/s a 2240 MB/s para un tamaño de mensaje de 128 KB, se podría decir que esta arquitectura corresponde a la arquitectura ReDMAC, propuesta en el Art.9 que consiste en una versión mejorada de un controlador DMA que se complementa en primer lugar con la arquitectura utilizada en el Art19 para mejorar la transmisión de grandes tamaños de datos, ya que el bus FIFO utilizado en otras arquitecturas tiene retrasos cuando tiene mucha capacidad, y los datos se transfieren cuando el buffer está lleno, además se complementa con la arquitectura propuesta por el Art04 debido a que en cada periférico se pueden emitir transacciones de

lectura y escritura utilizando uno o varios canales para hacer el acceso a la memoria, donde se implementa un microcontrolador completo por su rápida funcionalidad de comunicación haciendo que disminuyan las puertas de embarque en la configuración del canal además de permitir un bajo consumo de energía y el diseño de aplicaciones de baja potencia.

CONCLUSIONES

El uso del lenguaje VHDL en conjunto con los dispositivos FPGA resulta en una gran combinación ya que estos dispositivos ofrecen una gran flexibilidad al momento de diseñar un DMA como se ha podido evidenciar ene los estudios seleccionados en esta investigación.

Para el desarrollo de algoritmos de comunicación entre la unidad central de procesamiento y el DMA es preciso implementar la técnica de preconfiguración que se menciona en el Art06 donde se trata de ocultar la latencia de las comunicaciones de entrada/salida de los dispositivos que permite que este sistema reconfigurable resulte atractivo para el procesamiento de datos hiperespectrales con los que se requiere trabajar.

El uso del Acceso Directo a la Memoria (DMA), es esencial en las arquitecturas actuales de computadoras ya que permite a dispositivos de diferentes velocidades comunicarse sin someter al CPU a una carga masiva de interrupciones, las transferencias DMA son esenciales para aumentar el rendimiento de aplicaciones que requieran muchos recursos, además de la interacción de hardware, el acceso directo a memoria puede ser utilizado para descargar costosas operaciones de memoria, tales como copias de gran tamaño u operaciones de dispersión-reunión, desde la CPU a un motor de acceso directo a memoria dedicada

RECOMENDACIONES

Se recomienda tener muy claro el tipo de transacción que se desea realizar y sobre todo el tipo de datos que se va a trasmitir, dependiendo de estos se podrá elegir el modelado y diseño del DMA más adecuado a los requerimientos establecidos.

REFERENCIAS BIBLIOGRÁFICAS

Ammendola, R., Biagioni, A., Frezza, O., Cicero, F. Lo, Lonardo, A., Paolucci, P. S., … Vicini, P. (2013). Virtual-to-Physical address translation for an FPGA-based interconnect with host and GPU remote DMA capabilities. FPT 2013 - Proceedings

of the 2013 International Conference on Field Programmable Technology, 58–65. https://doi.org/10.1109/FPT.2013.6718331

Aswal, D., Singh, K., & Yadav, V. (2015). DESIGNING OF DMA CONTROLLER USING VHDL. 1(12), 608–612.

Bernardi, P., Cantoro, R., Gianotto, L., Restifo, M., Sanchez, E., Venini, F., & Appello, D. (2017). A DMA and CACHE-based stress schema for burn-in of automotive microcontroller. LATS 2017 - 18th IEEE Latin-American Test Symposium. https://doi.org/10.1109/LATW.2017.7906767

Biglari, M., Qasemi, E., & Pourmohseni, B. (2013). Maestro: A high performance AES encryption/decryption system. Proceedings - 17th CSI International Symposium on Computer Architecture and Digital Systems, CADS 2013, 145–148. https://doi.org/10.1109/CADS.2013.6714255

Didariya, A., Jasrotia, H. S., Gupta, R., Gurjar, S., & Tripathi, M. N. (2016). International Journal of Current Engineering and Technology Design and Implementation of Generic DMA using Vhdl. 856| International Journal of Current Engineering and Technology, 6(3), 856–861. Retrieved from http://inpressco.com/category/ijcet

Enami, T., Kawakami, K., & Yamazaki, H. (2015). DMA-driven control method for low power sensor node. 2015 IEEE Topical Conference on Wireless Sensors and Sensor Networks, WiSNet 2015, 53–55. https://doi.org/10.1109/WISNET.2015.7127418

Fjeldtvedt, J., & Orlandić, M. (2019). CubeDMA – Optimizing three-dimensional DMA transfers for hyperspectral imaging applications. Microprocessors and Microsystems, 65, 23–36. https://doi.org/10.1016/j.micpro.2018.12.009

Gonzalez, C., Lopez, S., Mozos, D., & Sarmiento, R. (2015). FPGA Implementation of the HySime Algorithm for the Determination of the Number of Endmembers in Hyperspectral Data. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 8(6), 2870–2883. https://doi.org/10.1109/JSTARS.2015.2425731

Google Académico. (n.d.). Retrieved August 21, 2020, from https://scholar.google.com.ec/

Home | Microsoft Academic. (n.d.). Retrieved August 21, 2020, from https://academic.microsoft.com/home

IEEE Xplore. (n.d.). Retrieved August 21, 2020, from https://ieeexplore.ieee.org/Xplore/home.jsp

Ji, F., Aji, A. M., Dinan, J., Buntinas, D., Balaji, P., Thakur, R., … Ma, X. (2012). DMA-assisted, intranode communication in GPU accelerated systems. Proceedings of the 14th IEEE International Conference on High Performance Computing and Communications, HPCC-2012 - 9th IEEE International Conference on Embedded Software and Systems, ICESS-2012, 461–468. https://doi.org/10.1109/HPCC.2012.69

Kavianipour, H., Muschter, S., & Bohm, C. (2014). High performance FPGA-based DMA interface for pcie. IEEE Transactions on Nuclear Science, 61(2), 745–749. https://doi.org/10.1109/TNS.2014.2304691

Khedkar, A. A., & Khade, R. H. (2017). High speed FPGA-based data acquisition system. Microprocessors and Microsystems, 49, 87–94. https://doi.org/10.1016/j.micpro.2016.11.006

Kitchenham, B. (2004). Procedures for Performing Systematic Literature Reviews. Joint Technical Report, Keele University TR/SE-0401 and NICTA TR-0400011T.1, 33.

Li, W., Zhao, Y., Liu, Y., & Chen, M. (2018). SMEFF: A scalable memory extension fabric for FPGA. 2017 International Conference on Field-Programmable Technology, ICFPT 2017, 2018-January, 40–47. https://doi.org/10.1109/FPT.2017.8280119

Li, Y., Cai, D., & Xu, Y. (2018). Improved DMA Algorithm for the PXIE Bus. (38), 1–5. https://doi.org/10.1145/3271553.3271591

Mantovani, P., Cota, E. G., Pilato, C., Di Guglielmo, G., & Carloni, L. P. (2016, October 13). Handling large data sets for high-performance embedded applications in heterogeneous systems-on-chip. 1–10. https://doi.org/10.1145/2968455.2968509

Morales, H., Duran, C., & Roa, E. (2019). A Low-Area Direct Memory Access Controller Architecture for a RISC-V Based Low-Power Microcontroller. 2019 IEEE 10th Latin American Symposium on Circuits and Systems, LASCAS 2019 - Proceedings, 97–100. https://doi.org/10.1109/LASCAS.2019.8667579

Mukherjee, S., Costa, F., Paul, R., Chakrabarti, A., Khan, S. A., Mitra, J., & Nayak, T. (2016). An efficient approach to evaluate PCIe DMA design and DMA performance for Common Readout Unit (CRU). Retrieved from https://inis.iaea.org/search/search.aspx?orig_q=RN:48040449

Ng, H. C., Choi, Y. M., & So, H. K. H. (2013). Direct virtual memory access from FPGA for high-productivity heterogeneous computing. FPT 2013 - Proceedings of the 2013 International Conference on Field Programmable Technology, 458–461. https://doi.org/10.1109/FPT.2013.6718414

Nguyen, H. K., Dong, K. P., & Tran, X. T. (2019). A reconfigurable multi-function dma controller for high-performance computing systems. NICS 2018 - Proceedings of 2018 5th NAFOSTED Conference on Information and Computer Science, 344–349. https://doi.org/10.1109/NICS.2018.8606841

RefSeek - Academic Search Engine. (n.d.). Retrieved August 21, 2020, from https://www.refseek.com/

Rota, L., Caselle, M., Chilingaryan, S., Kopmann, A., & Weber, M. (2015a). A new DMA PCIe architecture for Gigabyte data transmission. 2014 19th IEEE-NPSS Real Time Conference, RT 2014 - Conference Records. https://doi.org/10.1109/RTC.2014.7097561

Rota, L., Caselle, M., Chilingaryan, S., Kopmann, A., & Weber, M. (2015b). A PCIe DMA Architecture for Multi-Gigabyte per Second Data Transmission. IEEE Transactions on Nuclear Science, 62(3), 972–976. https://doi.org/10.1109/TNS.2015.2426877

ScienceDirect.com | Science, health and medical journals, full text articles and books. (n.d.). Retrieved August 21, 2020, from https://www.sciencedirect.com/

Scopus preview - Scopus - Welcome to Scopus. (n.d.). Retrieved August 21, 2020, from https://www.scopus.com/home.uri

Shanehsazzadeh, F., & Sadri, M. S. (2017). Area and performance evaluation of central DMA controller in Xilinx embedded FPGA designs. 2017 25th Iranian Conference on Electrical Engineering, ICEE 2017, 546–550. https://doi.org/10.1109/IranianCEE.2017.7985100

Tiwari, A., & Advanced, T. (2011). AMBA DEDICATED DMA CONTROLLER WITH. 4(1), 285–288.

Tumeo, A., Monchiero, M., Palermo, G., Ferrandi, F., & Sciuto, D. (2008). Lightweight DMA management mechanisms for multiprocessors on FPGA. Proceedings of the International Conference on Application-Specific Systems, Architectures and Processors, 275–280. https://doi.org/10.1109/ASAP.2008.4580191

Vanita, V. S. (2004). Direct Memory Access (DMA) Controller. Internatonal Journal of Innovative Research in Technology, 1(June), 1069–1071.

Wang, Y., Wang, T., Zhou, P., & Wang, X. (2014). Design and implementation of a flexible DMA controller in video codec system. International Conference on Digital Signal Processing, DSP, 2014-January, 78–82. https://doi.org/10.1109/ICDSP.2014.6900804

Ye, X., Li, Y., Du, Y., & Cai, Z. (2017). Design of data transmission system for speed measurement radar between ARM and FPGA based on embedded Linux. 2016 IEEE International Conference on Signal and Image Processing, ICSIP 2016, 343–346. https://doi.org/10.1109/SIPROCESS.2016.7888281

Zhao, Y., Li, M., Zhang, Y., Lin, Q., & Chen, Z. (2017). Research on FPGA timing optimization methods with large on-chip memory resource utilization in PCIe DMA. 2016 CIE International Conference on Radar, RADAR 2016. https://doi.org/10.1109/RADAR.2016.8059429