Implementación de un algoritmo de aprendizaje de máquina para la optimización del sistema hardware en una FPGA

Silva Salinas, Marcelo Javier

Vicerrectorado de Investigación, Innovación y Transferencia de Tecnológica

Centro de Posgrado

Maestría en Electrónica y Automatización Mención Redes Industriales

Trabajo de titulación previo a la obtención del título de Magister en Electrónica y Automatización, Mención Redes Industriales

Ing. Acosta Núñez, Julio Francisco PhD

6 de Julio del 2023

Latacunga

URKUND

Reporte de verificación de contenido

Document Information

Analyzed document	TESIS MARCELO SILVA.pdf (D171444062)
Submitted	6/28/2023 2:06:00 PM
Submitted by	Juan Carlos Altamirano
Submitter email	jc.altamiranoc@uta.edu.ec
Similarity	1%
Analysis address	jc.altamiranoc.uta@analysis.urkund.com

Sources included in the report

SA	**Cots_MuñozPR4.2.pdf** Document Cots_MuñozPR4.2.pdf (D135811453)	1
SA	**Ariel_Marcos_Condo_Cosimulacion.pdf** Document Ariel_Marcos_Condo_Cosimulacion.pdf (D29485776)	1

Ing. Acosta Núñez, Julio Francisco

Director

C.C.: 0501519490

Vicerrectorado de Investigación, Innovación y Transferencia de Tecnología

Centro de Posgrados

Certificación

Certifico que el trabajo de titulación, "**Implementación de un algoritmo de aprendizaje de máquina para la optimización del sistema hardware en una FPGA.**" fue realizado por el señor **Silva Salinas, Marcelo Javier,** el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizado por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Latacunga, 06 de julio de 2023.

.....

Ing. Acosta Núñez, Julio Francisco

Director

C.C.: 0501519490

Vicerrectorado de Investigación, Innovación y Transferencia de Tecnología

Centro de Posgrados

Responsabilidad de Autoría

Yo **Silva Salinas, Marcelo Javier**, con cédula de ciudadanía N° 1804207262, declaro que el contenido, ideas y criterios del trabajo de titulación*:* **Implementación de un algoritmo de aprendizaje de máquina para la optimización del sistema hardware en una FPGA** es de mí autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Latacunga, 06 de julio de 2023

.....

Silva Salinas, Marcelo Javier

C.C.: 1804207262

Vicerrectorado de Investigación, Innovación y Transferencia de Tecnología

Centro de Posgrados

Autorización de Publicación

Yo **Silva Salinas, Marcelo Javier,** con cédula de ciudadanía N° 1804207262, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **Implementación de un algoritmo de aprendizaje de máquina para la optimización del sistema hardware en una FPGA** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Latacunga, 06 de julio de 2023

.....

Silva Salinas, Marcelo Javier

C.C.: 1804207262

Dedicatoria

El presente trabajo de titulación es dedicado a mi hijo Lucas Silva, por el cual siempre tendré las ganas de superarme y ser un ejemplo a seguir en su vida; a mi esposa por alentarme en cada cosa que logramos juntos; a mi madre y hermana por siempre ser un apoyo incondicional en cada paso y logro que he tenido y a mi padre que desde el cielo sé que guiara cada camino de mi vida hasta volvernos a encontrar, gracias a todos.

A mi maestros y amigos que he tenido el placer de conocer en el transcurso de mi carrera universitaria como de masterado, proceso de aprendizaje y anécdotas que quedaran marcadas durante toda mi vida de estudiante y profesional.

Agradecimiento

Al finalizar un trabajo de titulación, en este caso de maestría, siempre el análisis objetivo te indica que todo lo que se ha logrado hubiese sido imposible sin la participación de personas e instituciones que han facilitado las cosas para que este trabajo llegue a un feliz término.

Agradezco a todos los ingenieros y doctores que me impartieron sus conocimientos durante todo el proceso de maestría, de manera especial a mi tutor de tesis el Ing. Julio Acosta. PHD, por su apoyo y confianza en mi trabajo de titulación, su capacidad para guiar y por aportar con los medios suficientes para llevar a cabo todas las actividades propuestas durante el desarrollo de esta tesis.

Gracias a los amigos que se pudo formar y gracias querida Universidad de las Fuerzas Armadas Espe sede Latacunga por las anécdotas y conocimientos que pude adquirir durante mi etapa universitaria de grado y posgrado, siempre te llevare en el corazón querida Universidad.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Resumen

Una de las tecnologías que más ha evolucionado en los últimos años son los algoritmos de aprendizaje de máquina, debido en mayor medida a los avances tecnológicos registrados en la última década sobre la capacidad de las FPGA's usadas en este tipo de aplicaciones, así como la reducción en su coste y consumo energético, lo que permite la utilización a gran escala de redes de este tipo de dispositivos. Esta investigación ha implementado un algoritmo de hardware genético mediante una FPGA para determinar el sentido de un movimiento, en este caso ilustrando la solución a la detección del cambio de posición. Al mismo tiempo que, se compara el desempeño de la implementación de hardware con una de software de la misma tarea utilizando MATLAB. Los procesos que componen el algoritmo genético, se implementan a través de módulos de hardware, tales como aleatorización de la población inicial, módulos para la muestra de resultados, módulo de memoria normalizada, caché para módulo de memoria. Los resultados se presentan gráficamente en una matriz de leds. Toda la implementación del algoritmo se realiza en lenguaje VHDL y la placa de desarrollo MOJO V3, en la que se encuentra la FPGA XILINIX SPARTAN 6. Los resultados de las pruebas de rendimiento de implementación de hardware y software también se presentan en forma tabular Por lo tanto, este trabajo de investigación, tiene como objetivo avanzar en el conocimiento científico y tecnológico necesario para el diseño y desarrollo de modelos de aprendizaje en dispositivos hardware específicos (FPGA's), con el fin de permitir su utilización en aplicaciones de sistemas en tiempo real y redes de sensores. Para ello se investigan y desarrollan estrategias de diseño para este tipo de dispositivos que maximicen la eficiencia en la utilización de recursos.

Palabras clave: sistemas embebidos, algoritmo de aprendizaje de máquina, fpga, hardware

Abstract

One of the technologies that has evolved the most in recent years are machine learning algorithms, due to a greater extent to the technological advances registered in the last decade on the capacity of FPGAs used in this type of applications, as well as the reduction in its cost and energy consumption, which allows the large-scale use of networks of this type of device. This research has implemented a genetic hardware algorithm through an FPGA to determine the direction of a movement, in this case illustrating the solution to position change detection. At the same time, the performance of the hardware implementation is compared with a software implementation of the same task using MATLAB. The processes that make up the genetic algorithm are implemented through hardware modules, such as randomization of the initial population, modules for the sample of results, normalized memory module, cache for memory module. The results are graphically presented in a matrix of leds. The entire implementation of the algorithm is done in VHDL language and the MOJO V3 development board, on which the XILINIX SPARTAN 6 FPGA is located. The results of hardware and software implementation performance tests are also presented in tabular form. Therefore, this research work has the objective of advancing in the scientific and technological knowledge necessary for the design and development of learning models in specific hardware devices (FPGA's), in order to allow their use in applications of systems in real time and sensor networks. To this end, design strategies for this type of device that maximize efficiency in the use of resources are investigated and developed.

Key words: embedded systems, machine learning algoritnm, fpga, hardware

Capítulo I

Generalidades

Antecedentes

Una de las tecnologías que más ha evolucionado en los últimos años son los algoritmos de aprendizaje de máquina, debido en mayor medida a los avances tecnológicos registrados en la última década sobre la capacidad de las FPGA's usadas en este tipo de aplicaciones, así como la reducción en su coste y consumo energético, lo que permite la utilización a gran escala de redes de este tipo de dispositivos. (Ortega Zamorano, 2015)

En la actualidad muchas aplicaciones precisan dotar de un actuador en los nodos sensores que convierta una señal eléctrica en un movimiento físico para modificar el entorno según la información recibida. La programación tradicional de sensores/indicadores puede conducir a decisiones incorrectas cuando las condiciones ambientales evolucionan con el tiempo, por lo que es necesario cambiar o adaptar el proceso de toma de decisiones a las nuevas circunstancias.

Una motivación de este trabajo es de dotar de inteligencia a las redes de sensores en este tipo de escenarios. Este algoritmo consistirá en proporcionar cierta información adicional junto a las variables medidas para dar un soporte automatizado a la toma de decisiones y al procesamiento distribuido, aportando una respuesta variable en función del cambio producido. (SANDOVAL-RUIZ, 2019)

A pesar del gran avance tecnológico producido en los últimos años en el campo de la microelectrónica, que permite disponer actualmente de microcontroladores con una capacidad de cómputo y memoria muy elevados a precios asequibles, los recursos con que cuenta siguen siendo una importante limitación para la implementación de algoritmos de aprendizaje sofisticados sobre este tipo de dispositivos. Por ello es necesario diseñar sistemas de modelos

de aprendizaje de máquina eficientes para desarrollar redes de sensores inteligentes. Por lo tanto, este trabajo de investigación, tiene como objetivo avanzar en el conocimiento científico y tecnológico necesario para el diseño y desarrollo de modelos de aprendizaje en dispositivos hardware específicos (FPGA's), con el fin de permitir su utilización en aplicaciones de sistemas en tiempo real y redes de sensores. Para ello se investigan y desarrollan estrategias de diseño para este tipo de dispositivos que maximicen la eficiencia en la utilización de recursos. (K. Neshatpour, 2018)

Planteamiento del problema

En la naturaleza, existen muchos fenómenos que presentan un comportamiento caótico o que no tienen un orden definido. Básicamente, cada fenómeno se observa a través de una forma de onda caótica y se representa por una serie de tiempo. Las series de tiempo generadas por un sistema caótico[1] son sensibles a las condiciones iniciales, una pequeña perturbación altera de manera significativa al sistema, por lo tanto, la dificultad para poder estimar valores futuros de la serie de tiempo incrementa. En la literatura existen varios métodos para predecir una serie de tiempo caótica, como algoritmos genéticos, métodos basados en inteligencia computacional, etc. Por ejemplo, se han utilizado algoritmos basados en una biblioteca de patrones que contiene datos pasados de una serie de tiempo, esto para determinar si algunas enfermedades tienen incidencia caótica. (A. Dinu, 2020) . No obstante, en la mayoría de los casos, no existe suficiente información que pueda ayudar a la predicción, por lo tanto, el uso de una serie de tiempo permite modelar el comportamiento de estos sistemas e inferir en su comportamiento. Se ha demostrado que los algoritmos de machine learning pueden representar cualquier función no lineal con una alta precisión. (N. A. Hazari, 2019)

[1] En los últimos años, se ha observado un aumento en el estudio de la teoría del caos en el contexto de funciones reales de variable real en IA. Esta teoría, que fue introducida por May (974, 1976) y Oster (1976), es ampliamente utilizada en modelos discretos no lineales.

El diseño de algoritmos de aprendizaje de máquina se divide en dos categorías: software y hardware (A. Al-hyari, 2018) En un software ad hoc se entrena y simula algoritmos supervisados y no supervisados, en ordenadores secuenciales para emular su comportamiento con una flexibilidad inherente. Por otro lado, el diseño en hardware permite verificar el funcionamiento del algoritmo y aprovechar el paralelismo del hardware. La realización VHDL de un algoritmo de machine learning proporciona una alta velocidad en aplicaciones en tiempo real. (Bouganis, 2015). No obstante, este tipo de aplicaciones carece de flexibilidad para la modificación estructural y representan alto costo de recursos. Esta investigación pretende hacer uso de FPGA's, para solventar problemas en el diseño de algoritmos de aprendizaje de máquina debido a: (T. Sato, 2018)

1. Necesidad de proporcionar una amplia gama de compuertas lógicas.
2. Uso deficiente de recursos de la arquitectura y el algoritmo, teniendo paralelismo en su naturaleza. Por lo tanto, la necesidad de que el diseño lógico trabaje en paralelo.
3. Problemas con la implementación hardware, por la existencia de una amplia gama general y no específica de herramientas de diseño, programación y síntesis.

Justificación e importancia

En muchos de los campos de la ciencia se ha generalizado la utilización de algoritmos de aprendizaje de máquina, y en todos ellos van aparecido nuevas aplicaciones donde la utilización de la programación tradicional sobre ordenadores convencionales no es suficiente para poder implementar de manera eficiente una solución a un problema dado, en este contexto el proyecto de investigación busca una alternativa a la implementación hardware de algoritmos de aprendizaje de máquina en plataformas de desarrolló con chipset FPGA's.

Las técnicas de diseño empleadas, hacen un énfasis especifico, en mejorar problemas creados ya sea por el elevado tiempo de cómputo de los ordenadores convencionales en

problemas complejos o por su consumo y dimensiones en sistemas empotrados, esto definido como el paradigma de esta investigación.

Dando paso a un sistema útil para los sistemas en tiempo real y las redes de sensores como dos exponentes de aplicaciones donde la utilización de machine learning, debe realizarse sobre dispositivos y con programación diferente a los convencionales, como las FPGA, que resultan más eficientes a la hora de una implementación de un modelo de aprendizaje de máquina.

Objetivo general del proyecto

Implementar un algoritmo de aprendizaje de máquina, como un dispositivo de detección inercial para la optimización del sistema hardware en una FPGA.

Objetivos específicos del proyecto

- Estudiar un algoritmo de aprendizaje de máquina, con la finalidad de evaluar las fortalezas y eficiencias de éste, para su diseño en los dispositivos hardware a implementar en la FPGA.
- Desarrollar estrategias de diseño del algoritmo de aprendizaje de máquina para reducir los recursos empleados en su implementación, sin alterar la integridad del algoritmo y modificar la complejidad de los procedimientos.
- Implementar de forma eficiente en una FPGA, el algoritmo de aprendizaje de máquina.
- Analizar los tiempos empleados en el proceso de aprendizaje, así como de la explotación del modelo.
- Evaluar el algoritmo de aprendizaje de máquina sobre una placa con FPGA, para la comparación de tecnologías con las firmas de archivos sobre predicción de posición, la gestión de alarmas de emergencia y los sensores de caídas.

Hipótesis de investigación

La implementación de un algoritmo aprendizaje de máquina en una FPGA, permitirá identificar y pronosticar movimientos inerciales para la optimización del sistema hardware.

Categorización de las variables de investigación

En consecuencia, de la hipótesis planteada se identifican las siguientes variables:

Variable Independiente: Implementación de un algoritmo de aprendizaje de máquina

Variable Dependiente: Optimización del sistema Hardware

Tabla 1

Operacionalización de las variables

VARIABLES	TIPO	DEFINICIÓN CONCEPTUAL	DEFINICIÓN OPERACIONAL	DIMENSIONES	INDICADORES
Optimización del sistema Hardware	Dependiente	La optimización de recursos de hardware se da mediante técnicas adecuadas de diseño con el uso eficiente de estructuras para configurar los algoritmos en el hardware	La optimización de los recursos mediante algoritmos de aprendizaje de máquina.	Bases de datos	Rangos de posición.
Implementación de un algoritmo de aprendizaje de máquina	Independiente	Los algoritmos de aprendizaje de máquina tienen la capacidad de identificar patrones en datos y así elaborar predicciones.	La variable posición es medida y manipulada mediante algoritmos de aprendizaje de máquina.	Aceleración de hardware y software	Parámetros de aceleración y tiempo de recursos usados por hardware y software

Capítulo II

Fundamentación teórica y referencial

Algoritmos de aprendizaje de máquina

El machine learning se encuentra en el campo de la inteligencia artificial que proporciona a los sistemas la capacidad de aprender y mejorar de manera automática, a partir de la experiencia, transforman datos en información, y con esta información pueden tomar decisiones. Para que un modelo realice predicciones de manera robusta y eficiente, necesita alimentarse de datos para comenzar el proceso de aprendizaje. Este proceso, llevado a cabo por un algoritmo, analiza y explora datos en búsqueda de patrones ocultos y el resultado de este aprendizaje es una función que opera sobre los datos para calcular una determinada predicción. (Zhou, 2021)

Además, se centra en el desarrollo de programas informáticos capaces de generalizar comportamientos a partir de los datos recibidos, además de poder cambiar cuando se exponen a nuevos datos. Estos procesos incluyen el aprendizaje (adquisición de información y reglas para el uso de la misma), el razonamiento (usando las reglas para llegar a conclusiones aproximadas o definitivas) y la autocorrección, estos algoritmos son de gran utilidad en varios dominios de aplicación, en grandes bases de datos que contienen regularidades implícitas, en dominios que son poco entendidos y donde no se posee el conocimiento necesario para desarrollar algoritmos eficaces, y en dominios donde los programas se deben adaptar en forma dinámica para responder a condiciones cambiantes en el ambiente (Ray, 2019).

Las aplicaciones de la inteligencia artificial incluyen asistencia sanitaria para hacer diagnósticos mejores y más rápidos, a través de aplicar el aprendizaje de la máquina con datos de pacientes y otras fuentes de datos disponibles. El aprendizaje automático o Machine Learning es una aplicación de la Inteligencia Artificial IA basada en la idea de dar a las

máquinas acceso a los datos y dejarles aprender por sí mismas, es decir, algoritmos que hacen a las máquinas más inteligentes (G. Meena, 2020).

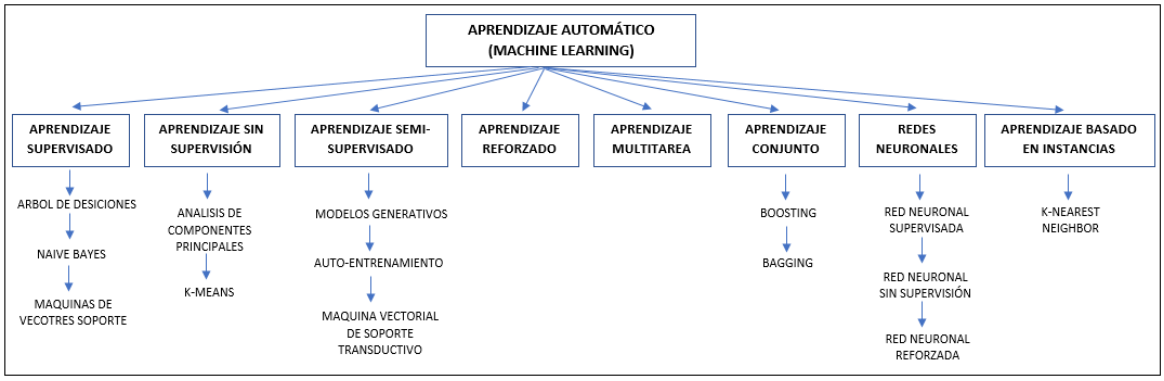
El aprendizaje automático se basa en diferentes algoritmos para resolver problemas de datos, no se evidencia ningún tipo de algoritmo único que sea mejor para resolver un problema. El tipo de algoritmo que se emplea depende del problema que se desea resolver, el número de variables y el tipo de modelo que conviene más.

Tipos de aprendizaje automático

Dependiendo de los datos disponibles y la tarea que queramos abordar, podemos elegir entre distintos tipos de aprendizaje automático.

Figura 1

Tipos de aprendizaje automático



Aprendizaje supervisado

El aprendizaje por clases (Ray, 2019), también conocido como aprendizaje supervisado, es aquel donde el objetivo es aprender un patrón a partir de un conjunto de datos que se utilizan de entrenamiento, y que permitirá realizar predicciones de conjuntos de datos no observados previamente. De esta forma, se puede indicar que el concepto de “supervisado” proviene de que, a partir de datos usados de ejemplo, que se usan como entrada para entrenar el modelo, se puede inferir cuál es la salida deseada.

Aprendizaje no supervisado

Para el aprendizaje sin clase (Yağanoğlu, 2020) también encontrado o conocido como aprendizaje no supervisado, se debe manejar datos sin etiquetar o datos sin estructura de los cuales no se tiene información previa que ayude a identificar patrones o saber a qué clase pertenecen los datos. Así este tipo de aprendizaje trata de crear distintos subgrupos o subconjuntos a partir de una exploración del conjunto de datos indicado, basándose por ejemplo en la similitud de las características (Li, 2020).

Un ejemplo de aplicaciones usando este tipo de aprendizaje se tiene la agrupación de genes y proteínas con similar funcionalidad o la agrupación de documentos para poder explorarlos de una forma más rápida, como por ejemplo Google Images.

Aprendizaje semi-supervisado

A medida que se fue profundizando en el estudio de las técnicas de aprendizaje automático, se comenzó a investigar un tercer tipo, conocido como aprendizaje semi-supervisado, este se encuentra a medio camino entre los dos tipos anteriormente descritos. Trata con conjuntos de datos donde se proporciona una información extra de la variable objetivo, como en el caso supervisado, pero no necesariamente para todas las instancias. (Albalate, 2013)

El aprendizaje semi-supervisado, debido a su estructura, se puede dividir principalmente en dos tipos, según el objetivo del análisis que se busque hacer a la base de datos. Por un lado, se tiene el aprendizaje transductivo, introducido por Vapnik, que separa el conjunto dado en conjunto de entrenamiento (donde la variable objetivo es conocida), y conjunto test donde esta no es dada. El objetivo de este tipo es tratar de predecir los valores de la variable objetivo del conjunto test. Por otro lado, el aprendizaje inductivo trata de obtener una función de predicción usando todo el conjunto {(X, Y)}, es decir, usando tanto los datos donde la variable objetivo es conocida como aquellos en los que no. (Zhu, 2008)

Aprendizaje reforzado

El aprendizaje reforzado (I. C. Freeman, 2018). El objetivo de este tipo de aprendizaje es conseguir un sistema que mejore su aprendizaje a medida que el entorno que lo rodea va cambiando o, dicho de otra forma, es extraer qué acciones deben ser elegidas en los diferentes estados para maximizar la recompensa. Así si la clasificación se realizó correctamente entonces ganará confianza dicha clasificación a través de una función de recompensa. De esta forma se pueden usar técnicas de exploración de un conjunto de estados y a través de prueba-error producir la salida adecuada habiendo obtenido la mejor recompensa posible. Como ejemplo en este tipo de aprendizaje se tiene el Alpha Go, desarrollado por Google Deep Mind, cuando en 2015 fue la primera máquina en ganar en el juego de mesa Go a un jugador profesional (O. Elíasson, 2019).

Aprendizaje multitarea

El aprendizaje multitarea (MTL) se ha utilizado con éxito en todas las aplicaciones de aprendizaje automático, desde el procesamiento del lenguaje natural y el reconocimiento de voz hasta la visión artificial y el descubrimiento de fármacos. MTL se presenta de muchas formas: aprendizaje conjunto, aprender a aprender y aprender con tareas auxiliares son solo algunos nombres que se han utilizado para referirse a él. En general, tan pronto como se encuentra optimizando más de una función de pérdida, está realizando efectivamente un aprendizaje de tareas múltiples (en contraste con el aprendizaje de una sola tarea). (Collobert, 2008)

Aprendizaje conjunto

El aprendizaje conjunto es un término general para los métodos que combinan múltiples inductores para tomar una decisión, generalmente en tareas de aprendizaje automático supervisado. Un inductor, también conocido como aprendiz base, es un algoritmo que toma un conjunto de ejemplos etiquetados como entrada y produce un modelo (por ejemplo, un

clasificador o regresor) que generaliza estos ejemplos. Usando el modelo producido, se pueden dibujar predicciones para nuevos ejemplos sin etiquetar. Un inductor de conjunto puede ser cualquier tipo de algoritmo de aprendizaje automático (por ejemplo, árbol de decisión, red neuronal, modelo de regresión lineal, etc.). La premisa principal del aprendizaje conjunto es que, al combinar múltiples modelos, los errores de un solo inductor probablemente serán compensados por otros inductores y, como resultado, el rendimiento global de predicción del conjunto sería mejor que el de un solo inductor. (Sagi, 2018)

Redes Neuronales

Se trata de un tipo de proceso de machine learning llamado aprendizaje profundo, que utiliza los nodos o las neuronas interconectados en una estructura de capas que se parece al cerebro humano. Crea un sistema adaptable que las computadoras utilizan para aprender de sus errores y mejorar continuamente

Una red neuronal artificial es un modelo computacional, inspirado en las redes neuronales humanas, que se utiliza para resolver una amplia variedad de tareas que son difíciles para la denominada “programación estructurada, las neuronas (o unidades de cálculo) son los bloques básicos de construcción de una red neuronal. Cada una de estas unidades tiene un número k de conexiones de entrada (inputs) y puede tomar cierto número de estados. (Fernandez, 2021)

Aprendizaje basado en instancias

Los algoritmos de aprendizaje basados en instancias; el clasificador del vecino más cercano es el algoritmo basado en instancias más conocido, estas son técnicas que, a diferencia del resto de técnicas de aprendizaje automático, no construyen un clasificador durante su fase de entrenamiento. Este tipo de clasificadores han heredado muchos rasgos beneficiosos, incluyendo su relativa simplicidad, su adaptabilidad a diferentes problemas generales, su flexibilidad a la hora de ser optimizado, la no necesidad de un proceso de

entrenamiento y la capacidad de poder incorporar nuevo conocimiento al clasificador de forma sencilla. (FERNÁNDEZ, 2018)

Sistemas neurocomputacionales en tiempo real

Los sistemas neurocomputacionales han tenido una gran repercusión en el ámbito científico desde su creación, pero no fue hasta los años 80 y principios de los 90 cuando se produjo un importante avance gracias en su mayor parte al desarrollo de la red de Hopfield, y en especial al algoritmo de aprendizaje de retro propagación (BackPropagation) ideado por Rumelhart y McLellan en 1986 que fue aplicado en el desarrollo de los perceptrones multicapa. Durante estos años se dedicaron grandes esfuerzos a la implementación hardware de sistemas neurocomputacionales sin demasiado éxito. La principal razón de este fracaso se puede atribuir a que la mayoría del trabajo realizado estaba dedicado a la implementación sobre circuitos específicos con tecnología ASIC. El rendimiento obtenido no fue competitivo para justificar la realización de circuitos específicos a gran escala, con lo que las implementaciones hardware sobre ASIC se descartaron en estos inicios.

En aquellos momentos la tecnología FPGA no estaba lo suficientemente desarrollada para ser una opción competitiva a la hora de realizar implementaciones neuro computacionales (Castaño Romero, 2016).

Sensores inteligentes

Los sensores (o detectores) son dispositivos que permiten la medición de variables químicas o físicas y las transforman en señales eléctricas, que son transmitidas por diferentes medios. Estos dispositivos son unidades autónomas que constan de un microcontrolador, una fuente de energía (usualmente una batería), un transmisor/receptor y un elemento sensor (Duangsuwan, 2018).

Los sensores inalámbricos pueden formar un grupo de sensores con ciertas capacidades sensitivas y de comunicación inalámbrica los cuales permiten formar redes sin infraestructura física preestablecida ni administración central (T. Cultice, 2020).

Unidad de medida inercial

La unidad de medida inercial (IMU) es un dispositivo microelectromecánico relacionado con magnitudes físicas inerciales. Estos detectores, anclados a un punto de acción, son capaces de detectar, por ejemplo, la aceleración que ocurre a lo largo de tres direcciones perpendiculares entre sí, lo que da lugar a un sistema de coordenadas propio del sensor.

En los últimos tiempos, las IMU's se han hecho popular en el mercado por la miniaturización del dispositivo, tal es por su bajo consumo de energía, bajo costo y su peso ligero. Se puede encontrar a veces que las IMU's vienen integradas, es decir varios subsistemas unidos entre sí. Uno de esos son los algoritmos de calibración cuya función es afinar más las medidas, reduciendo así los errores de cualquier tipo al mínimo (Korkishko Y. N., 2018).

Historia del Hardware Evolutivo

Uno de los mayores logros de la tecnología es el computador, el ser humano medio posee en su casa una multitud de objetos dotados de un procesador como: el ordenador, el celular, el equipo de música, la lavadora entre otras que son ejemplos de dispositivos programables. Un dispositivo programable basa su versatilidad en el software que permite que una sola máquina pueda desarrollar un rango de tareas grandes y con una facilidad asombrosa.

El hardware es otra historia debido a que es algo estático, inmutable, que siempre hará aquello para lo que el fabricante lo diseño. Nadie puede "crear hardware" a menos que se trate de algo sencillo o que posea de un complejo equipo de fabricación de circuitos integrados, solo disponible para grandes empresas fabricantes de hardware.

De esta forma con la aparición en el mercado de dispositivos de lógica programable (PLD) y mas tarde de los field programmable gate array (FPGA), nace el hardware reconfigurable. A diferencia con el hardware programable este puede reorganizar su configuración interna (su circuito) para dar lugar a circuitos digitales totalmente distintos de acuerdo a las necesidades. Una FPGA moderna con altas prestaciones, puede convertirse en un procesador, una unidad de punto flotante, una unidad de procesado de gráficos o puede integrar diversos elementos.

Las FPGAs pretenden aprovechar todo su potencial de forma que mientras una parte tiene su configuración fija que lleva a cabo el proceso principal y otras complejas, otra parte se dedica a implementar en hardware aquellas tareas que llevarían demasiado tiempo el procesador, pero ofreciendo la posibilidad de ser reconfigurada de acuerdo a las necesidades para que el hardware sea reutilizado.

Algoritmos evolutivos

Los algoritmos evolutivos hacen énfasis a la naturaleza, la vida es posible gracias a la capacidad de las poblaciones de seres vivos de adaptarse al entorno.

La teoría evolutiva se basa en:

- Las especies evolucionan con el paso de las generaciones dando lugar a individuos diferentes.
- La evolución se da en incrementos pequeños, por ende, los hijos son similares a sus padres, pero con ciertas diferencias.
- Los individuos que mejor se adaptan al medio son los que sobreviven mas tiempo de tal forma tienen más descendencia.
- Tras muchas generaciones una especie puede haber cambiado lo suficiente para adaptarse al medio en el que se encuentra, llegando a dar a nuevas especies.

- Los productos artificiales y los programas y equipos electrónicos “evolucionan” de manera distinta:
- Algunos productos aparecen de la nada en el mercado y se basan en una reinvención de otros productos existentes.
- Otros productos surgen de la modificación de productos existentes y en particular es fruto de un elaborado estudio que ha determinado que dicha modificación debía hacerse.
- El proceso de evolución y evaluación de resultados no se produce automáticamente, sino que necesita de la intervención humana para que pueda llevarse a cabo.

Los algoritmos evolutivos parten de un elemento generado, aleatorio o bien artificialmente, donde le introducen modificaciones y evalúan el resultado mediante un método que indique numéricamente un factor de calidad de dicho resultado conocido como fitness (aptitud), para pretender automatizar este proceso de creación.

Si el resultado es peor, se vuelve al elemento original, si es mejor, se elimina el original y se mantiene el nuevo con el fin de realizar futuras modificaciones de dicho elemento hasta llegar a encontrar el óptimo. Este proceso de optimización de puede hacerse solo con único individuo original y uno nuevo, sino que siempre existe una población con un gran numero de individuos y se elige aleatoriamente entre ellos, ordenándolos y eliminando los peores cada vez y quizá también eliminando los viejos.

Algoritmos genéticos

Un método muy usado para implementar un algoritmo evolutivo es mediante programación genética, que consiste en descomponer modularmente el sistema a evolucionar de tal forma que pueda ser representado mediante un conjunto de parámetros. Estos parámetros son denominados gen, y el conjunto de todos estos que describe la configuración de un individuo en concreto recibe el nombre de cromosoma.

Son un tipo de algoritmos que diseñan, abstraen y regeneran casos relacionados con la reproducción natural. Producen cálculos inteligentes utilizando métodos aleatorios que reproducen los parámetros aplicados de selección genética de forma natural. En su forma más general, un algoritmo genético basado en poblaciones muestra un grupo inicial de posibilidad en las soluciones. Se pueden reproducir en secuencias iterativas, esta es la solución que se aplica a los sistemas, a estos procesos de iteraciones siguientes se le llama formación. En estas formaciones, los elementos pudieron medir el progreso utilizando medidas de posición. **Fuente especificada no válida.**.

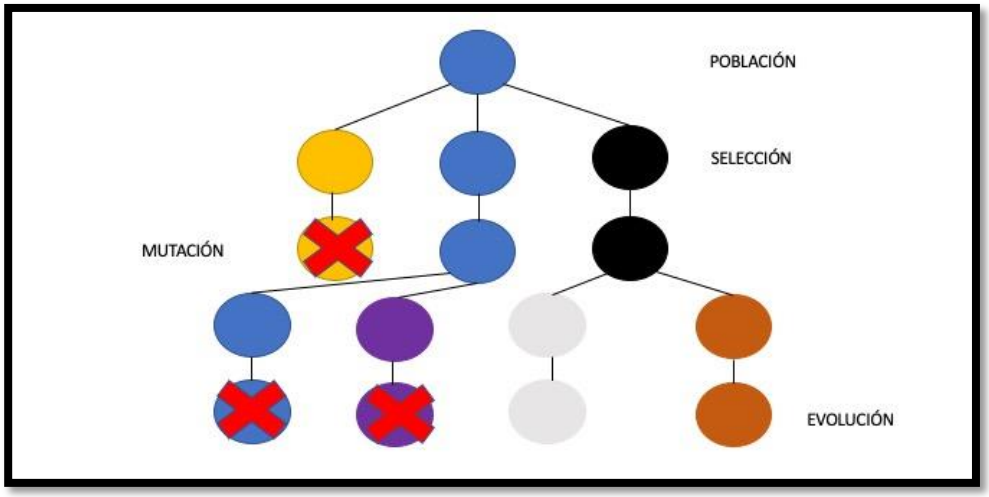
Conviene que el conjunto de todas las posibles combinaciones sea lo mejor posible, ya que un espacio de búsqueda demasiado grande puede hacer que el algoritmo genético no converja y no encuentre puntos óptimos debido a la gran cantidad de combinaciones inadecuadas que puede encontrar antes de hallar el de mejor utilidad.

El proceso de creación de nuevos cromosomas a partir de uno antiguo recibe el nombre de mutación. Se selecciona aleatoriamente un cierto numero de genes y se altera su valor obteniendo un nuevo cromosoma similar al anterior, pero con algunos cambios. Después se evalúa la calidad del individuo con el nuevo cromosoma empleando una función numérica, denominada fitness, y en función de estos resultados o bien se almacena o se descarta al individuo.

Teniendo en cuenta que el proceso de reproducción natural es biológico, debemos tener en cuenta que al implementar algoritmos genéticos a la solución de problemas es necesario seguir con una secuencia de ejecución. De esta manera uno de los factores importantes es que la población tenga una cantidad de datos nada despreciable, como para obtener una variedad de soluciones. Idealmente, la población debe generarse al azar para tener una variedad, cuando la cantidad de datos obtenidos no se genere al azar, considere que se garantiza la variedad en la población inicial, como se muestra en la Figura 2.

Figura 2

Esquema de interpretación de algoritmo genético básico



Los algoritmos genéticos son procedimientos de investigación avanzadas confiables para este caso en particular, donde el inconveniente inicial es encontrar caminos que no sean conflictivos y, por lo tanto, generar posibles arrays de soluciones, esto se considera como un parámetro del conjunto de decisión. Los parámetros para la interpretación del algoritmo genético básico se muestran en la Tabla 2.

Tabla 2

Parámetros de interpretación de algoritmo genético básico

PARÁMETRO	FUNCIÓN
$\boldsymbol{\mu}$	Índice de selección
$\boldsymbol{P}$	Array de posibilidades
$\boldsymbol{x, y, z}$	Identificadores de espacio $\mathbb{R}^3$
$\boldsymbol{l}$	Contador de posiciones del vector
$\boldsymbol{fe}$	Índice de evolución
$\boldsymbol{fs}$	Índice de selección
$\boldsymbol{fm}$	Índice de mutación
$\boldsymbol{D}$	Array de Posición
$\boldsymbol{a}$	Posición inicial del Array
$\boldsymbol{b}$	Posición final del Array

La población se define de forma matemática como:

$$(l) = \{D^{l}_{x,y,z}, \dots\dots\dots\dots, Dn^{ln}_{xn,yn,zn}\}; \qquad (2.1)$$

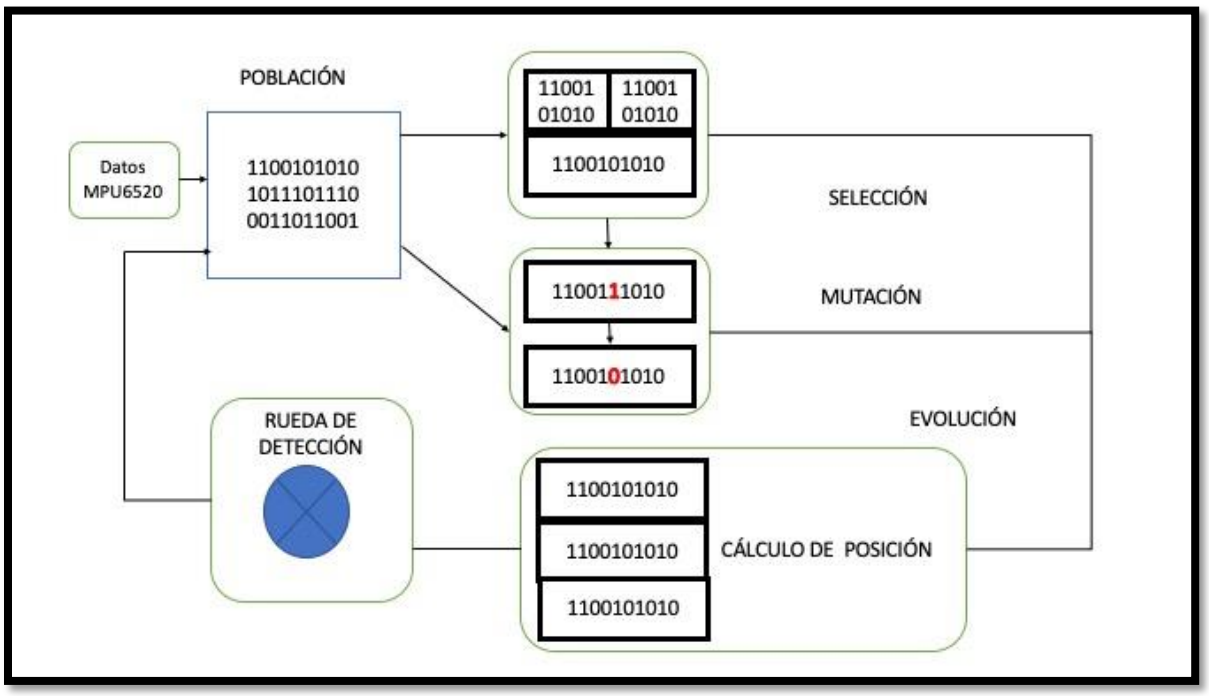
$$x, y, z \in \mathbb{R}^3 ;$$

$$l = 1, \dots\dots\dots, n ;$$

Mediante cada sucesión de l, para las coordenadas X_l, Y_l, Z_l , los cuales representan una posición dentro del espacio de referencia, datos que se recopilan del sensor mpu 6520; para formar la información de la población. Debido a esto cada coordenada se evalúa por funciones asociadas a la posibilidad de una probabilidad $D(D_x, D_y, D_z)$ como se ve en (2.1), las mismas que se asigna a una adaptación del módulo de selección adecuado.

Figura 3

Interpretación de algoritmo genético básico



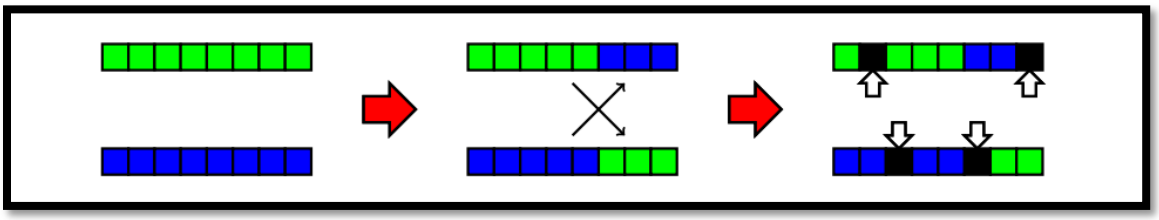
En la Figura 3, el sistema usa la población como punto de partida para manejar de forma sucesiva las posiciones recopiladas en el espacio de configuración, esto se realiza con el uso de procesos aleatorios, por ejemplo:

1. Determinación de población inicial preprocesada.
2. Aumento de grado en la posibilidad para evolucionar, mutar; de los elementos detectados en una población adecuada para identificar determinada posición.

Existen otros mecanismos de generación de nuevos cromosomas además de la mutación, uno de estos mecanismos es el de recombinación o crossover como se muestra en la Figura 4, que se genera un nuevo cromosoma a partir de segmentos de dos cromosomas diferentes con esto se pretende que el individuo resultante reúna las fortalezas de ambos padres o en el peor de los casos las debilidades, pero el algoritmo evolutivo se encarga de eliminar estos casos.

Figura 4

Mutación y crossover



El VRC (Virtual Reconfigurable Circuit)

A principios del año 2000 una propuesta permitió seguir avanzando en el campo del hardware evolutivo: el VRC o circuito virtual reconfigurable el cual es una disposición virtual que define la aplicación especifica de un circuito reconfigurable, en vez de trabajar con la complejidad de la configuración de una FPGA en esta se usan estándares de descripción HDL y contiene (Sekanina):

- Un vector o array de CFBs, bloques del sistema que actúan como nodos del array, cada nodo contiene todas las funciones necesarias dentro de los elementos de procesamiento que realizan operaciones sencillas dentro del sistema evolutivo.
- Conjunto de multiplexores funcionales (uno por cada CFB) para seleccionar la función requerida del PE.
- Conjunto de multiplexores que proveen la conexión entre los nodos.
- Un registro extenso que trabaja como memoria de la configuración.

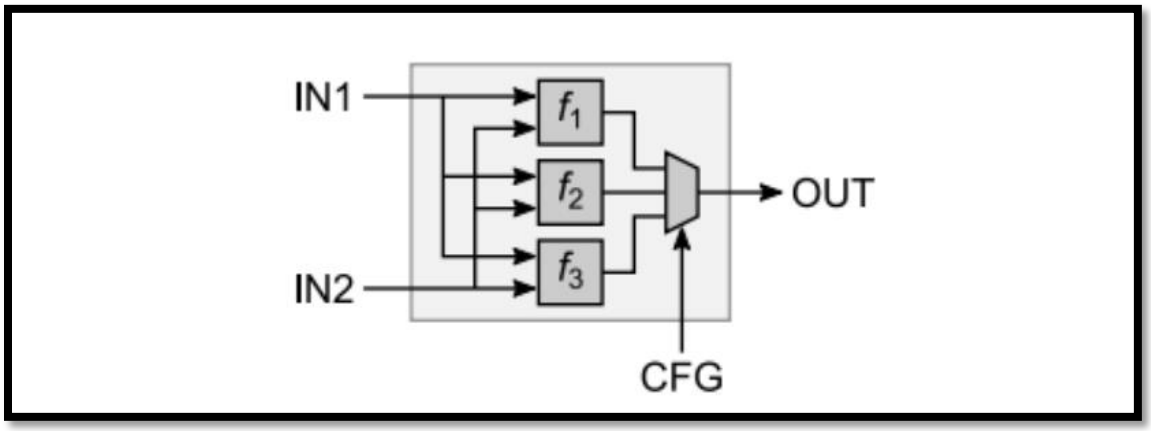
La reconfiguración del circuito virtual reconfigurable es bastante rápida ya que solo escribe en el registro que indica la señal de selección de los multiplexores. Esta capa virtual genera un elevado consumo de recursos porque produce una sobrecarga por las implementaciones simultaneas de cada función de cada nodo del array junto con las modificaciones de las conexiones de los multiplexores, lo que provoca retrasos que reducen la frecuencia máxima de trabajo.

Los VRC se estudiaron como una posibilidad para implementar los CGP (Cartesian Genetic Programming) el cual es un tipo de estructura evolutiva que se había planteado para la evolución de las FPGAs. Este CGP se describe como un circuito digital con nodos de procesamiento que se van mapeando y conectando entre ellos. Con cartesiano se refiere a las coordenadas para localizar en dos dimensiones cada nodo dentro de una malla. Se usa con una técnica simple de evolución con pequeñas poblaciones de un padre y entre 4 y 10 hijos, modificando pequeñas partes del sistema, es considerado el único mecanismo evolutivo ya que la recombinación no parece tener un resultado efectivo (F. Miller, 2011).

En la Figura 5, se puede visualizar un ejemplo de representación virtual, un nodo de tres funciones y el multiplexor que luego selecciona que función se va a utilizar (Xilinx, 2014).

Figura 5

Esquema de un nodo de sistema evolutivo.



Sistemas evolutivos basados en FPGAs

En los años 2000 Xilinx introdujo el ICAP Internal Configuration Access Port (Puerto de Acceso a la Configuración Interna), que permitió la auto reconfiguración nativa, al proveer una vía de acceso a la reconfiguración de la FPGA desde la propia FPGA. Este puerto, que no se encontraba en los sistemas basados en multiplexores se implementó con el objeto de poder generar circuitos auto reconfigurables como un apoyo para el Hardware Evolutivo.

Sin embargo, la velocidad de reconfiguración era demasiado baja como para implementar aplicaciones basadas en hardware, problemas como la falta de herramientas para DPR, los aumentos de tamaño de los bitstreams y sus limitaciones, problemas que siguen sin estar resueltos en la actualidad (Javier Mora).

Asimismo, el desarrollo de la tecnología de Hardware Evolutivo se vio bloqueado durante los primeros años 2000, ya que no se consiguió mejorar la tecnología que lo soportaba.

A finales de los años 2000 se desarrolló el PCAP, Processor Configuration Access Port (Puerto de Acceso al Procesador), para la reconfiguración desde procesadores ARM embebidos, permitiendo generar sistemas de auto reconfiguración heterogéneos que ofrecían nuevas posibilidades además de la mejora de velocidad de reconfiguración (Javier Mora).

En la Tabla 3, se muestra la evolución de las diferentes aplicaciones que se han ido realizando con sistemas de Hardware Evolutivo en los últimos años.

Tabla 3

Caso de aplicación en distintas tecnologías.

Application	Reconfiguration	FPGA	EA	Fitness
FIR filters	Register values	Any, not reported	HW	HW
Image filters	VRC	XC2V3000	HW	HW
Hash functions	VRC	XC4VFX20	HW	HW
RBN[a]/Cellular automaton	LUT-DPR (ICAP)	Virtex II	MicroBlaze	MicroBlaze
Image filters	VRC	XC2VP50	PowerPC	HW
Sonar spectrum classifier	VRC	XC2VP30	PowerPC	HW
Arith. circuits	VRC	XCV2000E	HW	2×HW
CGP accelerator	VRC	XC2VP50	PowerPC	HW
Face image classif.	VRC	XC2VP30	MicroBlaze	HW
Image filters, classif.	VRC	XCV2000E	HW	HW
Multiple Constant Multipliers[b]	VRC	XC2VP50	HW	HW
Face classifiers	SRLUTs[c]	XC2VP	PowerPC/MicroBlaze	HW
CGP accelerator	VRC	XC5VFX100T	PowerPC	4×HW
Small comb. circuits	Module&LUT-DPR	Virtex 4	PowerPC	HW
Classifiers and comb. circuits	Module-DPR	Virtex 4	PowerPC	HW
Same as [34]	SRLUTs-CFGLUT5[d]	Virtex 5	MicroBlaze	HW
Image filters	VRC & Module-DPR[e]	Zynq-7000	ARM Cortex-A9	HW
Image filters	Module-DPR	Virtex 5	MicroBlaze	HW
Face/sonar classif.	LUT-DPR[f] (ICAP)	Virtex	MicroBlaze	HW
Image filters	VRC & LUT-DPR (PCAP)	Zynq-7000	ARM Cortex-A9	HW
Image filters	VRC & LUT-DPR (PCAP)	Zynq-7000	ARM Cortex-A9	6×HW
Image filters	LUT-DPR (ICAP)	Virtex 5	MicroBlaze	8×HW

En la primera columna se observa la aplicación, como se fue avanzando desde reconfiguración VRC con algoritmo evolutivo en hardware hasta sistemas híbridos reconfigurables con auto reconfiguración por microprocesador. En la segunda columna se muestra el tipo de sistema que se usó para la reconfiguración, ya sea con VRC o módulos adiciones. En la tercera columna el modelo de FPGA que se utilizó para implementar el sistema. En la cuarta el dispositivo encargado de la reconfiguración, procesadores internos o externos. Y en la última columna la plataforma encargada de la función, una función

fundamental para los algoritmos evolutivos que calcula lo bien o mal que se comporta el sistema (Fernández).

Se realizaron varios prototipos que contenían algoritmo evolutivo vía hardware (cuarta columna), sin embargo, se obtenían mejores resultados cuando era un procesador el que se encargaba del algoritmo evolutivo, debido a que este simplifica en gran medida el sistema y no perjudicaba la velocidad del circuito. Se implementó de manera más seguida este algoritmo en microprocesadores, cuando la tecnología permitió implementar procesadores internos con el desarrollo del PCAP, encontrando ejemplos que usan el Microblaze para la reconfiguración.

Es necesario resaltar que el cálculo de los valores que toma la función “fitness” (la función principal de la evolución) consume gran cantidad de tiempo, debido a esto no se consiguen grandes mejoras de tiempo en el algoritmo evolutivo, esta es la razón por la que esta función si suele implementarse vía hardware.

En cuanto a las aplicaciones de sistemas basados en VRC, se encuentran filtros de imágenes, circuitos de lógica combinacional, clasificadores de paquetes y multiplicadores de constantes múltiples, entre muchas otras (J. Torresen, 2008).

Capítulo III

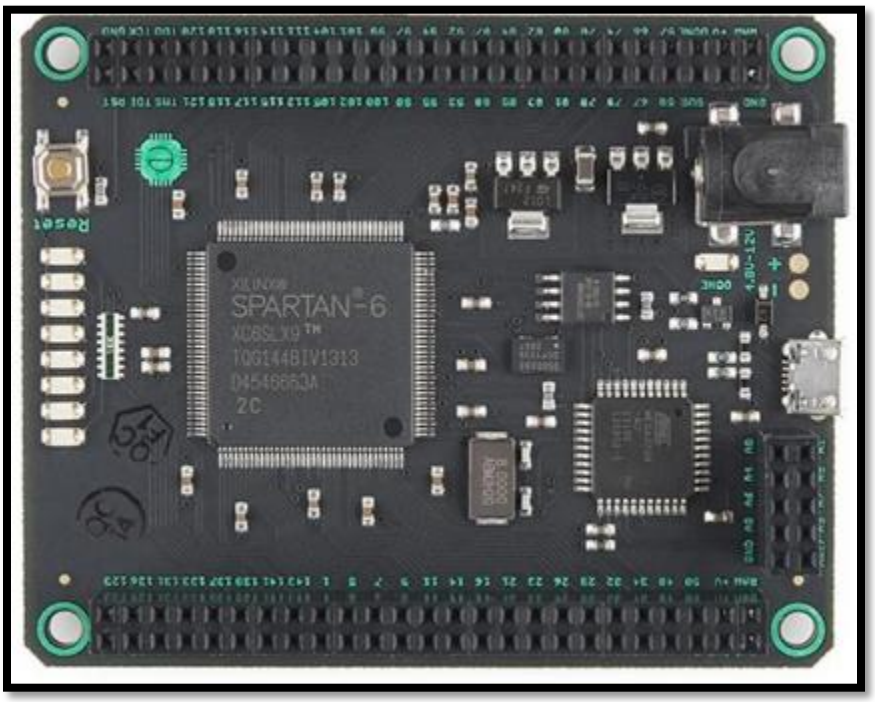
Implementación

Implementación en hardware

La implementación del algoritmo genético se realizó empleando una tarjeta de desarrollo MOJO V3 de ALCHINTRY, la cual contiene un FPGA Spartan 6 de XILINIX así como diversos bloques de memoria, puertos de comunicación, un indicador de posición mediante un array de leds. El software empleado para configuración y comunicación con la tarjeta es proporcionado por XILINIX.

Figura 6

Hardware para implementar el algoritmo genético, ALCHINTRY MOJO V3.



La implementación del algoritmo genético se realiza utilizando la placa de chasis ALCHINTRY MOJO V3, que contiene la FPGA XILINIX Spartan 6, así como múltiples bloques de memoria, puertos de comunicación e indicación de posición a través de una serie de luces

LED. El software de desarrollo para comunicación además de configuración de la FPGA es proporcionado por XILINIX(AMD). En este contexto el lenguaje utilizado para programar además de configurar las funcionalidades necesarias para el algoritmo en la FPGA es VHDL.

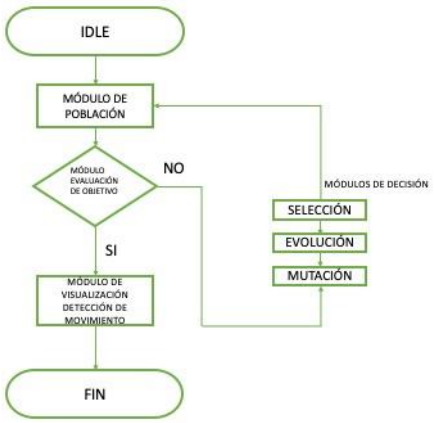
VHDL es un lenguaje de hardware para diseñar sistemas electrónicos. Fue desarrollado con la supervisión directa del Instituto de Ingenieros Eléctricos y Electrónicos (IEEE), se utiliza como guía para el lenguaje VHDL estándar, el estándar IEEE 1076 de 1987. Al ser un estándar IEEE, es posible que tenga una revisión periódica.

Dentro de la investigación, se tiene el módulo generador de población inicial, Módulo de decisión de objetivo, Módulo de decisión (Selección, Evolución, Mutación), Módulo de visualización de detección de movimiento, que integran el algoritmo a implementar.

En la Figura 7, se puede visualizar el diagrama esquemático de implementación del algoritmo genético.

Figura 7

Interpretación de algoritmo genético básico



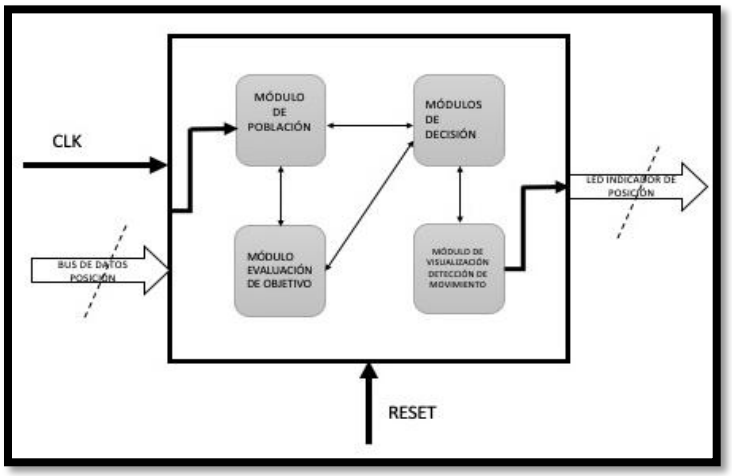
Se muestra la conexión de los módulos que lo integran. Estos módulos se asignan a las memorias relacionadas con los bloques de la RAM interna que tiene la FPGA, cabe mencionar que no se usan los módulos de memoria externos que también tiene la tarjeta, sino que solo los

recursos internos de la FPGA. En el esquema de la Figura 8, se puede identificar las líneas de entrada y salida a la implementación de la arquitectura las cuales se muestran a continuación:

- RESET Señal de operación reset que proviene de un pulso externo a la tarjeta que es la base de la inicialización y puesta en marcha de la operación en la implementación.
- CLK Señal de reloj de 50MHz que es provista por la tarjeta de desarrollo. Usada para generar los procesos requeridos en el funcionamiento del algoritmo.
- BUS DE DATOS POSICIÓN: Son los datos recopilados del sensor mpu 6520 el cuál envía coordenadas en x,y,z.
- LED INDICADOR DE POSICIÓN: Array de leds que permite visualizar un cambio en la posición del sistema con eso se puede verificar el funcionamiento del algoritmo.

Figura 8

Interpretación en bloques del algoritmo genético básico



En este contexto se incluyen ciertas señales de funcionamiento como la señal de inicio que proviene de un interruptor o de una señal interna que provee la tarjeta para inicio de forma manual el inicio del algoritmo. Así también, la generación de la línea de entrada de interruptores de la tarjeta de desarrollo, que indica el estado de implementación desplegado en la barra de

leds. Cuando la línea tiene el valor de cero, los leds con los bits “0”, “1”; indican “Fin”, y cuando la línea se encuentra en uno, en todos los leds identifica que se encontró la solución.

Operación del algoritmo genético en hardware

El algoritmo inicia cuando la señal de entrada (de nivel de voltaje alto), activa la funcionalidad del módulo de población. Por esto, se describe la operación de todos y cada uno de los módulos como se muestra en la Tabla 4.

Tabla 4

Descripción del Módulo de Población.

Paso	Acción
1	Generador de Población inicial
2	Termina la población inicial, inicia la evaluación
3	Después inicia la selección
4	Realizada la selección, procede la evolución.
5	Luego de la evolución, inicia mutación
6	Finalizada la mutación de la población inicial, ejecuta nuevamente la evaluación

Funcionamiento síncrono de los módulos del algoritmo. La función a realizar es la obtención de señales de reseteo de memoria que sean configuradas en los módulos apropiados según la detección de posición. Después de recibir la señal inicial, el algoritmo comienza a funcionar, por lo que genera las señales de manera uniforme para realizar los procesos en los módulos y también controla las generaciones que se aplica dicho algoritmo. El algoritmo final es responsable de activar la visualización de los resultados en el array de Leds.

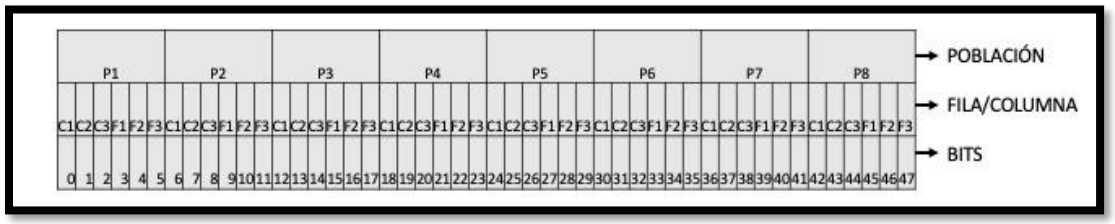
Módulo de Población

Este módulo genera aleatoriamente una población de decisión inicial de grupos de individuos, de 48 bits cada uno, almacenados en una memoria para indicar la cantidad de población. Cada coordenada se codifica como se muestra en la Figura 9. Se sugirió utilizar una

población de 64 elementos porque se requiere un contador de mayor cantidad de bits para manipular una cantidad superior a 80 muestras, para utilizar una cantidad más manejable en cuestión de procesamiento con dimensiones de datos cercanas a un dato real se usa una población inicial de 64 coordenadas.

Figura 9

Descripción de la codificación de datos a memoria de población



De la Figura 9, cada conjunto de población con 6 bits para codificación, 3 columnas (C) y 3 filas (F), que indican la posición de cada población en una matriz de 8 C y 8 F. La población inicial se obtiene mediante un proceso de sensado de muestras que utiliza una base de tiempo diferente a la base de tiempo de la operación del algoritmo, los datos de 48 bits son proporcionados por contadores independientes de 8 bits que comienzan con diferentes valores.

Los contadores se detienen cuando se obtiene datos para ubicarlos en la memoria de población, hasta completar datos de las posiciones. La frecuencia de reloj utilizada en esta aleatorización y todas las aleatorizaciones requeridas para ejecutar el algoritmo es de 50 MHz, tomada de un generador ubicado en la misma placa de desarrollo.

Módulo de Evaluación de Objetivo

Este módulo implementa una función que determina el número de movimientos bruscos que ocurren en cada decisión de la población. Este se calcula por partes, los ataques se obtienen comparando directamente las columnas y filas de cada población, luego el número de diagonales con pendiente positiva que se obtiene al sumar las columnas y filas de cada

población, compara directamente el número total de piezas de cada población además si son iguales, se contará en los movimientos.

Por último, las diagonales con pendiente negativa se obtienen restando fila a fila de cada población, es directamente comparable con la resta de cada población, si resultan ser iguales se cuenta el ataque, el punto.

Los datos resultantes de evaluar los movimientos en las coordenadas que se almacenan en la memoria del evaluador, corresponden a la dirección de memoria de evaluación, lugar donde es almacenada la decisión. Por lo tanto, cada solución en población de memoria tiene evaluación en una similar dirección. Al tener un número nulo de selecciones, el proceso se para, indicando a través de la señal junto con la dirección de la memoria de población donde se encuentra la solución.

Módulo de Decisión

Se realiza con el método de enfrentamiento, que parte de determinar qué parejas tienen los mejores resultados, es decir, menores fallas al detectar cambios de posición. Al seleccionar el par de soluciones de inicio en la memoria de población, se usa una dirección aleatoria entre 0 y 63. De la dirección se busca un par de similar memoria, con la igual dirección se obtendrá el número de ataques del par en la memoria de evaluación.

La dirección del par y el número de cambios que se almacenan en la memoria asociada a la decisión, este proceso vuelve a comenzar con la segunda dirección de memoria para la población hasta que se cubren las direcciones. La encriptación de los datos almacenados en la memoria de decisión es de 12 bits, correspondiendo los primeros 5 bits al número de cambios del par y los bits restantes correspondientes a sus direcciones.

Para elegir, pasa por la memoria de la decisión y la memoria de elección, compara el número de cambios, si es mayor, se procede a reemplazar la decisión de la memoria de población por su contraparte y el proceso finaliza después 48 intentos de negociar los

protocolos de evaluación y selección. Es importante indicar que la nueva ejecución resultante actuará como padre durante la evolución.

Etapa de Evolución.

La etapa implementa la evolución por decisión (inicio) porque la selección se almacenó previamente en la memoria de la población, datos con el resultado del proceso evolutivo es que los descendientes se ubican en la memoria de la población, ahora se convierte en población infantil.

Durante este proceso, se obtienen números aleatorios correspondientes a la posibilidad de evolución además de la puntuación de evolución. Probabilidad obtenida de un contador de ejecución libre de 0 a 100, puntuación de evolución obtenida de otro contador de ejecución libre de 0 a 47, cada respuesta tiene una longitud aproximada de 48 bits.

La posibilidad de que se elija la evolución es superior al 80%, siendo el operador de evolución uno de los primordiales, y en la práctica la probabilidad definida suele elegirse entre el 50% y el 90%. Junto con el criterio de evolución se forma un control de evolución a 48 bits, compuesta por ceros desde el bit menos significativo al más rango del 0 a 47; Esta palabra tiene su negación.

El proceso evolutivo es el siguiente: se revisan dos padres cercanos de memoria de la población, cada padre tiene pares de registros; usando la función AND con control y registre el otro caso usando la función AND con control negativa. Luego, el primer hijo tiene un OR entre los registros 1 y 2 (a,b), el segundo hijo tiene un OR entre los registros 2 y 1(b,a) Este proceso da como resultado que dos hijos reemplacen a sus padres y sean nuevamente almacenados en sus respectivos domicilios en la memoria de la población. Este proceso comienza de nuevo por pares de padres hasta cubrir toda la memoria de la población.

Etapa de Mutación.

La etapa depende del proceso de variación de cada individuo nacido debido al proceso evolutivo, que queda registrado en la memoria relacionada a la población. Cada dato es leído considerando paso a paso de 0 a 47. De cada bit transmitido se tiene una posibilidad de evolución aleatoria, si esta es menor o igual al 6% se realiza una mutación de un bit dado incluyendo la negación del bit especificado; si la posibilidad es mayor, entonces no se realiza la negación y se transmite el nuevo bit.

La etapa tiene lugar en todos los niños de la población de memoria. La posibilidad de mutación aleatoria obtenida de un contador de ejecución libre oscila entre 0 y 100. Hay que tener en cuenta que el tiempo base del contador es físicamente distinto del tiempo base en la que trabaja el algoritmo.

La posibilidad de mutación se establece en función de la longitud de la selección, definida como 1/L, donde L corresponde a 48 y es aproximadamente 6% de las mutaciones, por lo que cada individuo mutante tiene una variable media de un bit.

Módulo de Visualización de detección de movimiento

La funcionalidad de este módulo comienza cuando el módulo de revisión activa la señal de visualización, lo que significa que ha encontrado una solución que no genera fallas en la detección y también envía la dirección de la solución específicamente a través del bus desde la dirección relacionada a la solución de la memoria de la población, resuelve el problema y lo envía al módulo de control de Leds en el bus. La señal también es recibida por el módulo de selección para representar la solución. La generación especificada se mostrará en un LED de estado dentro de la tarjeta.

Capítulo IV

Resultados y pruebas experimentales

Configuración del experimento

El programa genético paralelo fue probado en dos arquitecturas. La primera es una plataforma FPGA basada en dos dispositivos MOJO V3, cada una de las cuales cuenta con arreglos de hardware. La segunda plataforma es una CPU con un procesador de un microcontrolador.

Sobre estas dos plataformas se ejecutó el programa genético para 4, 8 y 12 variables, diferentes tamaños de poblaciones y diferente número de hilos o procesos e islas ejecutados.

El tiempo de respuesta del algoritmo ejecutado en el microcontrolador, con 1 y 2 CPU para poblaciones de 1024 o 32786 individuos y diferente cantidad de islas, depende del tamaño del problema debido a que complejidad de los individuos crece exponencialmente con la cantidad de variables del problema y estos son evaluados por software. En contraste el tiempo de respuesta no tiene una gran dependencia del tamaño de problema ya que se evalúa en hardware.

También se observa que para evaluar el AG con un tamaño de población grande (32.768 individuos) tiene mejor desempeño en 2 CPU que en una sola. Sin embargo, al aumentar la población de 1024 a 32.768 el tiempo de respuesta no se incrementa en la misma proporción ya que el tiempo empleado para las comunicaciones está incluido en los dos escenarios.

Caso práctico del experimento

Dentro de los experimentos posibles para un adecuado análisis, elegimos el problema one-max para evaluar el rendimiento del sistema. El tamaño de la población (n) y el cromosoma longitud (l) son 256 y 32 respectivamente. Uno se compara millones de generaciones. En la Tabla 5 se presenta una comparación entre la versión de software (ESP8266) y hardware (MOJOV3). La versión de software es escrita en lenguaje C++ y compilado usando el compilador gcc. El software se ejecuta en ESP8266 de 160 MHz, NonSDK. El resultado muestra que el hardware es 1000 veces más rápido que el software que se ejecuta en una estación de trabajo. El número de generaciones ejecutadas en hardware se da mediante la ecuación (4.1):

$$\frac{frecuencia\ de\ operación}{reloj\ utilizado\ por\ genereación} \tag{4.1}$$

$\frac{frecuencia\ de\ operación}{reloj\ utilizado\ por\ genereación} = \frac{16M}{3} = 5.3$millones de generaciones por segundo.

Se puede identificar que el hardware rinde mucho más rápido por mayor dimensión y velocidad.

Tabla 5

Caso de comparación en rendimiento entre Hardware y Software.

Software (160 MHz Esp8266)	Hardware (FPGA 32 MHz)	Acelerar
2:30 minutos	0,15 seg.	1,000

Los datos de rendimiento de los algoritmos genéticos basados en hardware (MOJO V3) y software (ESP8266), propuestos son presentado en la Tabla 5. De esta forma, en la última columna, las aceleraciones se basan en el cálculo realizado en Matlab mediante computador que se dan por este caso práctico. Al comparar el trabajo con otros estudios realizados es por lo general difícil; ya que en su totalidad utilizan diferentes algoritmos, los problemas de optimización y medidas de aceleración por ende variaran de forma impredecible. Los trabajos iniciales en este

campo se basan en un caso simple de Algoritmo Genético (AG). Debido a que el denominado Simple AG, no es adecuado para la implementación de hardware, los trabajos recientes recurren al uso de Steady-state (Estado estacionario) AG.

Con AG de estado estacionario, se puede lograr la aceleración extrema. En el trabajo, se evalúa a un individuo cada ciclo de reloj. No podemos afirmar con antelación que el rendimiento del hardware, es mejor que el otro (software), desde la ejecución; las evaluaciones son importantemente para la diferenciación. Por otro lado, en un caso extremo al acelerar por software revela su capacidad para diferenciar la captura de los beneficios de la implementación por VLSI.

A continuación, en Matlab se calcula las operaciones de las firmas para su comparación respecto a la aceleración y tiempos de ejecución; de esta forma se generaliza el rendimiento del hardware Compact AG a otros problemas de optimización. Dividimos la versión de software de Compact AG en dos partes: la parte de evaluación (parte uno) y la parte restante de cálculo del algoritmo (parte dos). En este caso t_{EV} es $\frac{A}{A+B}$ y t_{AG} es $\frac{B}{A+B}$, Donde A es el tiempo gastado en la parte uno y B es el tiempo usado en la parte dos. Dado que s_{EV} sea la aceleración del hardware sobre el software para la primera parte y sea s_{AG} la aceleración del hardware sobre el software para la segunda parte. Por lo tanto, al acelerar de hardware compacto de algoritmo genético para a problema particular se puede mostrar en la Ecuación (4.2).

$$acelhw = \frac{1}{\frac{t_{AG}}{s_{AG}} + \frac{t_{EV}}{s_{EV}}} \qquad (4.2)$$

La t_{AG}, t_{EV} y s_{EV} dependen del problema de optimización y de la función de aptitud. Además, la s_{EV} depende en gran medida de la rapidez con la que se pueda evaluar a un individuo en hardware. El s_{AG} es conocido ya que es independiente del problema. El s_{AG} puede obtenerse de la Ecuación (4.3).

$$s_{AG} = \frac{tiempo\ gastado\ en\ la\ parte\ dos(sW)}{tiempo\ gastado\ en\ la\ parte\ dos(HW)} \tag{4.3}$$

Consecuentemente,

$$s_{AG} = \frac{1.48\ min}{0.10\ sec} = 1.080$$

En la Ecuación (4.2), sustituimos s_{AG} con 1,080 y sustituimos t_{AG} con $1 - t_{EV}$

$$acelhw = \frac{1.080 s_{EV}}{s_{EV}(1-t_{EV})+1.080 t_{EV}} \tag{4.4}$$

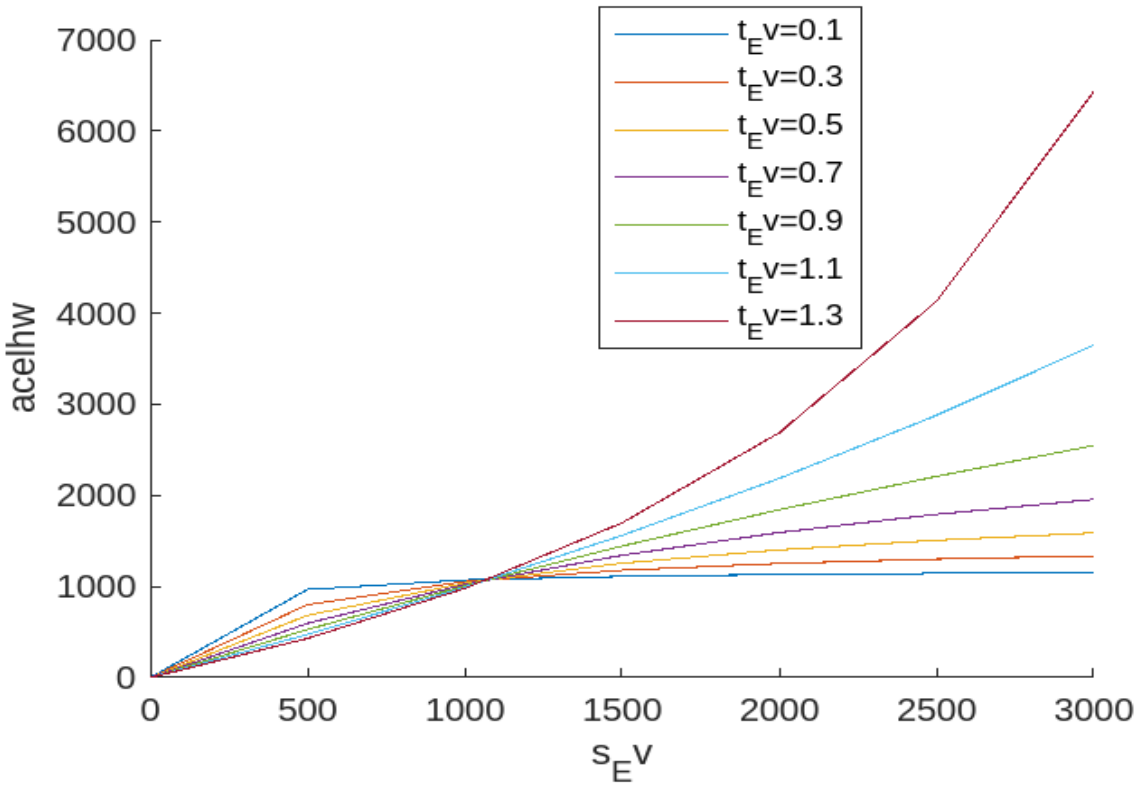
Aquí se logra la aceleración del hardware Compact AG. Cabe señalar que el aumento de velocidad en la Ecuación (4.4) se basa en un ESP8266 de 160 MHz.

La aceleración se muestra en la Figura 10. Cuando s_{EV} es menor que s_{AG} un problema cuyo tipo t_{EV} es menor se ejecuta más rápido. Esto va junto con la ley de Amdahl's desde la parte más lenta debe ser lo más pequeño posible.

Cuando s_{EV} es mayor que s_{AG}, la parte dos se convierte en un cuello de botella. Como resultado, un problema cuyo tipo es t_{EV} más grande, corre más rápido. Esto nos ayuda a verificar el adecuado desempeño de una arquitectura manejada por hardware.

Figura 10

Aceleración del hardware en el algoritmo genético.



Capítulo V

Conclusiones y recomendaciones

Conclusiones

La estrategia alternativa propuesta en este estudio se basa en la implementación de una plataforma híbrida, que combina el procesamiento en CPU y FPGA. Esto permite una mayor flexibilidad en la configuración de los parámetros del algoritmo evolutivo, ya que se pueden asignar tareas específicas a cada uno de los procesadores, según sus capacidades y limitaciones.

Por ejemplo, los cálculos más intensivos pueden ser asignados a la FPGA, que es una plataforma altamente paralela y especializada en el procesamiento de datos. Mientras tanto, las tareas menos intensivas pueden ser manejadas por la CPU, que es una plataforma más versátil y generalista.

Esta distribución de tareas permite una optimización más eficiente y precisa, ya que se aprovechan las fortalezas de cada plataforma para maximizar el rendimiento del algoritmo evolutivo. Además, la plataforma híbrida también permite una mayor capacidad de procesamiento y una respuesta más rápida, lo que es especialmente importante en aplicaciones donde se requiere una toma de decisiones en tiempo real.

Los resultados de los experimentos mostraron una aceleración superior a 41 evolucionando más de 32K individuos en una plataforma comparado con la plataforma CPU del microcontrolador.

Se demostró que para poblaciones pequeñas (1024 individuos) es menos eficiente ejecutarlo en la FPGA en comparación con ejecutarlo en la plataforma CPU ya que el tiempo empleado en las comunicaciones se vuelve importante.

En resumen, la estrategia alternativa propuesta en este estudio representa un avance significativo en el campo de los algoritmos evolutivos, al permitir una mayor eficiencia y flexibilidad en la evaluación de individuos sobre plataformas basadas en CPU y FPGA. Esto puede tener importantes aplicaciones en la optimización de sistemas complejos y en la toma de decisiones en tiempo real, lo que podría tener un impacto significativo en diversos campos, como la ingeniería, la robótica y la inteligencia artificial.

Recomendaciones

Utilizar herramientas de diseño avanzadas. Las herramientas de diseño de alto nivel como Verilog Hardware Description Language o VHDL pueden ayudarlo a diseñar e implementar algoritmos genéticos en FPGA de manera más eficiente.

Optimización del uso de recursos: Al implementar un algoritmo genético en una FPGA, es importante optimizar el uso de los recursos disponibles. Esto puede incluir optimizar el tamaño de los cromosomas y elegir la arquitectura FPGA adecuada.

Utilizar técnicas de paralelización: los algoritmos genéticos se pueden paralelizar fácilmente, lo que significa que se pueden implementar en FPGA utilizando técnicas de paralelización para mejorar el rendimiento y la eficacia.

Considere el uso de bibliotecas IP: las bibliotecas de propiedad intelectual (IP) pueden proporcionar bloques de ingeniería previa que se pueden usar para implementar algoritmos genéticos FPGA más rápidos y eficientes.

Realice pruebas exhaustivas. Al implementar un algoritmo genético en un FPGA, es importante realizar pruebas exhaustivas para garantizar que el diseño sea robusto y funcione correctamente. Esto puede incluir pruebas en emuladores y en el propio FPGA.

Bibliografía

A. Al-hyari, Z. A. (2018). An Effective FPGA Placement Flow Selection Framework using Machine Learning. *30th International Conference on Microelectronics (ICM)*, 164-167.

A. Dinu, G. M. (2020). Debug FPGA projects using machine learning. *International Semiconductor Conference (CAS)*, 173-176.

Adobe. (2021). *Substance 3D*. Recuperado el 10 de 9 de 2021, de Substance Painter: https://www.adobe.com/la/products/substance3d-painter.html

Albalate. (2013). *Semi-supervised and unsupervised machine learning.*

Alfaro Ruiz, V. (2002). Métodos de sintonización de controladores PID que operan como reguladores. *Ingeniería, Revista de la Universidad de Costa Rica Vol 12 (1-2)*, 21-36.

Alfaro, V. (2002). Métodos de sintonización de controladores PID que operan como reguladores. *Ingeniería, Revista de la Universidad de Costa Rica Vol 12 (1-2)*, 21-36.

Arranz de la Peña, J. &. (2013). *ALGORITMOS GENÉTICOS.*

Aström, K., & Hägglund, T. (2009). *Control PID Avanzado.* Madrid, España: Pearson, [Tabla], 17 de Septiembre de 2021 Recuperado de Control PID Avanzado.

Åström, K., & Hägglund, T. (2009). *Control PID Avanzado.* Madrid: Pearson Educación.

Aula.21. (2020). *Centro de Formación Técnica para la Industria*. Recuperado el 15 de 10 de 2021, de Instrumentación Industrial: todo lo que necesitas saber: https://www.cursosaula21.com/que-es-la-instrumentacion-industrial/

Aula21. (2020). *Centro de Formación Técnica para la Industria*. Recuperado el 2021 de 10 de 15, de Qué es el protocolo Ethernet Industrial: https://www.cursosaula21.com/que-es-ethernet-industrial/

AUTODESK. (2021). *AUTODESK*. Recuperado el 10 de 09 de 2021, de 3ds mAX: https://latinoamerica.autodesk.com/products/3ds-max/features

AUTODESK-Inc. (2021). *AUTODESK*. Recuperado el 10 de 9 de 2021, de AutoCAD Plant 3D:

https://latinoamerica.autodesk.com/products/autocad/included-toolsets/autocad-plant-3d?us_oa=dotcom-us&us_si=01a5081b-ea00-4496-9878-8b5c20643f85&us_st=AUTOCAD%20PLANT%203D&us_pt=PLNT3D

Bouganis, M. B. (2015). FPGA based nonlinear Support Vector Machine training using an ensemble learning. *25th International Conference on Field Programmable Logic and Applications (FPL)*, 1-4.

Castaño Giraldo, S. A. (s.f.). *Control Automatico Educación*. Recuperado el 25 de 07 de 2021, de Método de la Integral del Error Sintonia Controlador PID: https://controlautomaticoeducacion.com/control-realimentado/metodo-de-la-integral-del-error-sintonia-controlador-pid/ , [Tabla].

Castaño Romero, F. H. (2016). Castaño Romero, F., Haber Guerra, R. E., DInteligencia computacional embebida para la supervisión de procesos de microfabricación.

Castaño, S. (s.f.). *Control Automático Educación*. Recuperado el 25 de 09 de 2021, de Método de la Integral del Error Sintonia Controlador PID: https://controlautomaticoeducacion.com/control-realimentado/metodo-de-la-integral-del-error-sintonia-controlador-pid/

Charre-Ibarra, S., Alcalá-Rodríguez, J., & López-Luiz, N. (2014). Sistema didáctico de control de presión. *Formación Universitaria, vol. 7*, 33-40.

Collobert. (2008). Una arquitectura unificada para el procesamiento del lenguaje natural. Actas de la 25.ª Conferencia internacional sobre aprendizaje automático.

Creus, A. (2010). *Instrumentación Industrial* (Vol. Octava). México: Alfaomega.

Cultice, T. I. (2020). Smart home sensor anomaly detection using convolutional autoencoder neural network. *IEEE International Symposium on Smart Electronic Systems*, 67-70.

Duangsuwan, S. T. (2018). A study of air pollution smart sensors lpwan via nb-iot for thailand smart cities 4.0. . *10th International conference on knowledge and smart technology*, 206.

Emerson. (2021). *Emerson*. Recuperado el 3 de 10 de 2021, de Qué son los transductores?: https://www.emerson.com/es-es/automation/valves-actuators-regulators/controllers-instruments/transducers

Erosa, D. (10 de Junio de 2019). *OpenWebinars*. Recuperado el 05 de 10 de 2021, de Que es Unity?: https://openwebinars.net/blog/que-es-unity/

F. Miller, P. T. (2011). Cartesian Genetic Programming, in: Genetic NASA/ESA., (págs. 184–191).

Fernandez. (2021). Redes Neuronales.

Fernández, B. (s.f.). *Análisis y optimización de algoritmos genéticos para el filtrado de imágenes mediante hardware evolutivo.*

FERNÁNDEZ, G. (2018). Modelos híbridos de aprendizaje basados en instancias y reglas para Clasificación Monotónica.

Fredes, C. A., Hernández, J. P., & Díaz, D. A. (2012). Potencial y Problemas de la Simulación en Ambientes Virtuales para el Aprendizaje. *Formación universitaria, 5(1)*, 45-56.

G. Meena, D. S. (2020). Traffic Prediction for Intelligent Transportation System using Machine Learning. *3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things (ICETCE)*, 145-148.

I. C. Freeman, A. J. (2018). What are they Researching? Examining Industry-Based Doctoral Dissertation Research through the Lens of Machine Learning. *17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 1338-1340.

J. Torresen, G. A. (2008). Partial Reconfiguration Applied in an On- 40 line Evolvable Pattern Recognition System.

Javier Mora, R. S. (s.f.). *Javier MoOn the Scalability of Evolvable Hardware Architectures: Comparison of Systolic Array and Cartesian Genetic Programming.*

K. Kazumura, Y. A. (2021). Index of gait stability using inertial measurement unit. *IEEE 3rd Global Conference on Life Sciences and Technologies (LifeTech)*, 29-30.

K. Neshatpour, H. M. (2018). Design Space Exploration for Hardware Acceleration of Machine Learning Applications in MapReduce. *IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines*, 221-221.

Kinoshita, T., & Yamamoto, T. (2016). Design of a Data-Oriented Cascade Control System. *IEEJ Transactions On Electronics*, 703-709.

Korkishko, V. D. (2018). Use of Neural Networks in Q-Learning Algorithm. Computer and Information Sciences. 188–195.

Korkishko, Y. N. (2018). High-precision inertial measurement unit IMU-5000. *IEEE International Symposium on Inertial Sensors and Systems (INERTIAL)*, 1-4.

Lab-Volt. (2004). Fundamentos del Control de Procesos usando el programa LVPROSIM. *Instrumentación y control de procesos*, 1-1,1-2. doi:ISBN 2-89289-711-4

Li, M. L. (2020). Survey on lie group machine learning. *Big Data Mining and Analytics*, 235-258.

Lucidchart. (2021). *Lucidchart*. Recuperado el 06 de 10 de 2021, de Que son los diagramas de tuberías e instrumentación: https://www.lucidchart.com/pages/es/que-son-los-diagramas-de-tuberias-e-instrumentacion

M. Bhuyan, K. K. (2017). Nonlinear Mobile Link Adaptation Using Modified FLNN and Channel Sounder Arrangement. *IEEE Access*, 10390-10402.

MathWorks. (2021). *MATLAB*. Recuperado el 11 de 9 de 2021, de Matemáticas. Gráficas. Programación.: https://es.mathworks.com/products/matlab.html

MathWorks-Inc. (2021). *Centro de Ayuda*. Recuperado el 11 de 9 de 2021, de Descripción del producto MATLAB: https://la.mathworks.com/help/matlab/learn_matlab/product-description.html

Morales, R., & Ramirez, R. (2013). Sistemas de Control Moderno: VOLUMEN I: Sistemas de control continuo. *Editorial Digital tecnológico de Monterrey*, 29. Obtenido de http://prod77ms.itesm.mx/podcast/EDTM/ID295.pdf

N. A. Hazari, A. O. (2019). Analysis and Machine Learning Vulnerability Assessment of XOR-

Inverter based Ring Oscillator PUF Design. *IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS)*, 590-593.

O. Elíasson, C. W. (2019). A man-machine mind meld for quantum computing: Can an online game that combines human brainpower with AI solve intractable problems? *IEEE Spectrum*, 37-41.

Ogata, K. (2010). *Ingenieria de control moderna.* Madrid: Pearson,17 de Septiembre de 2021[Tabla] Recuperado de : Ingenieria de control moderna.

Ogata, K. (2010). *Ingeniería de control moderna* (Quinta ed.). Madrid: Pearson.

Ortega Zamorano, F. (2015). *Algoritmos de aprendizaje neurocomputacionales para su implementación hardware.* MÁLAGA: UNIVERSIDAD DE MÁLAGA.

Ortega, J., Vera, G., & Burgos, Á. (2003). La realidad virtual y sus posibilidades. *Etic@net*, 1-17. Obtenido de https://dialnet.unirioja.es/servlet/articulo?codigo=6871642

Ray, S. (2019). A Quick Review of Machine Learning Algorithms. *International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, 35-39.

S. Duangsuwan, A. T. (2018). A Study of Air Pollution Smart Sensors LPWAN via NB-IoT for Thailand Smart Cities 4.0. *10th International Conference on Knowledge and Smart Technology (KST)*, 206-209.

Sagi. (2018). Ensemble learning: A survey. Wiley Indisciplinary Reviews: Data Mining and Knowledge Discovery.

Sanchis, R., Ariño, C., & Romero, J. (2010). Automatización industrial. *Universitat Jaume ISBN: 978-84-693-0994-0*, 111.

SANDOVAL-RUIZ, C. (2019). VHDL Neural Control Model on FPGA technology oriented to Sustainable Applications. *Ingeniare. Revista chilena de ingeniería*, 383-395.

Schlumberger. (2021). *Oilfield Glossary en Español*. Recuperado el 18 de 10 de 2021, de separador horizontal: https://glossary.oilfield.slb.com/es/terms/h/horizontal_separator

schulumberger. (2021). *Oilfield Glossary en Español*. Recuperado el 18 de 10 de 2021, de

quemador: https://glossary.oilfield.slb.com/es/terms/f/flare

Sekanina, L. (s.f.). Virtual Reconfigurable Circuits for Real-World Applications of Evolvable Hardware, in: Evolvable Systems: From Biology to Hardware.

Şengör, E. E. (2015). A neurocomputational model implemented on humanoid robot for learning action selection. *International Joint Conference on Neural Networks (IJCNN)*, 1-6.

Smith, C., & Corripio, A. (1991). *Control Automático de Procesos Teoría y Práctica.* México: Limusa.

T. Cultice, D. I. (2020). Smart Home Sensor Anomaly Detection Using Convolutional Autoencoder Neural Network. *IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS*, 67-70.

T. Sato, S. C. (2018). Designing a Fine-Tuning Tool for Machine Learning with High-Speed and Low-Power Processing. *18th International Symposium on Communications and Information Technologies (ISCIT)*, 204-207.

Wolframalpha. (2021). *WolframAlpha Computational Intelligence*. Recuperado el 28 de 10 de 2021, de https://www.wolframalpha.com/input/?i2d=true&i=inverse+Laplace+transform+0.94726*%5C%2840%29Divide%5Bexp%5C%2840%29-2.3372s%5C%2841%29%2C1%2B17.636s%5D%5C%2841%29

WolframAlpha. (2021). *WolframAlpha Computational Intelligence*. Recuperado el 28 de 10 de 2021, de https://www.wolframalpha.com/input/?i2d=true&i=inverse+Laplace+transform+0.94726*%5C%2840%29Divide%5Bexp%5C%2840%29-2.3372s%5C%2841%29%2C1%2B17.636s%5D%5C%2841%29

Wonderware. (2021). *Wonderware*. Recuperado el 28 de 10 de 2021, de Interfaz Hombre-Máquina (HMI): https://www.wonderware.es/hmi-scada/que-es-hmi/

Xilinx, I. (2014). *Series FPGAs Configurable Logic Block (UG474).*

Yağanoğlu, S. K. (2020). An Example of Performance Comparison of Supervised Machine Learning Algorithms Before and After PCA and LDA Application: Breast Cancer Detection. *Innovations in Intelligent Systems and Applications Conference (ASYU)*, 1-6.

Zhou, Z. H. (2021). *Machine learning. Springer Nature.*

Zhu. (2008). *Semi-Supervised Learning Literature Survey.*

Anexos