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 estudios 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. Julio Francisco Acosta Núñez, PhD

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.

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.

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

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 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. 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.

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.

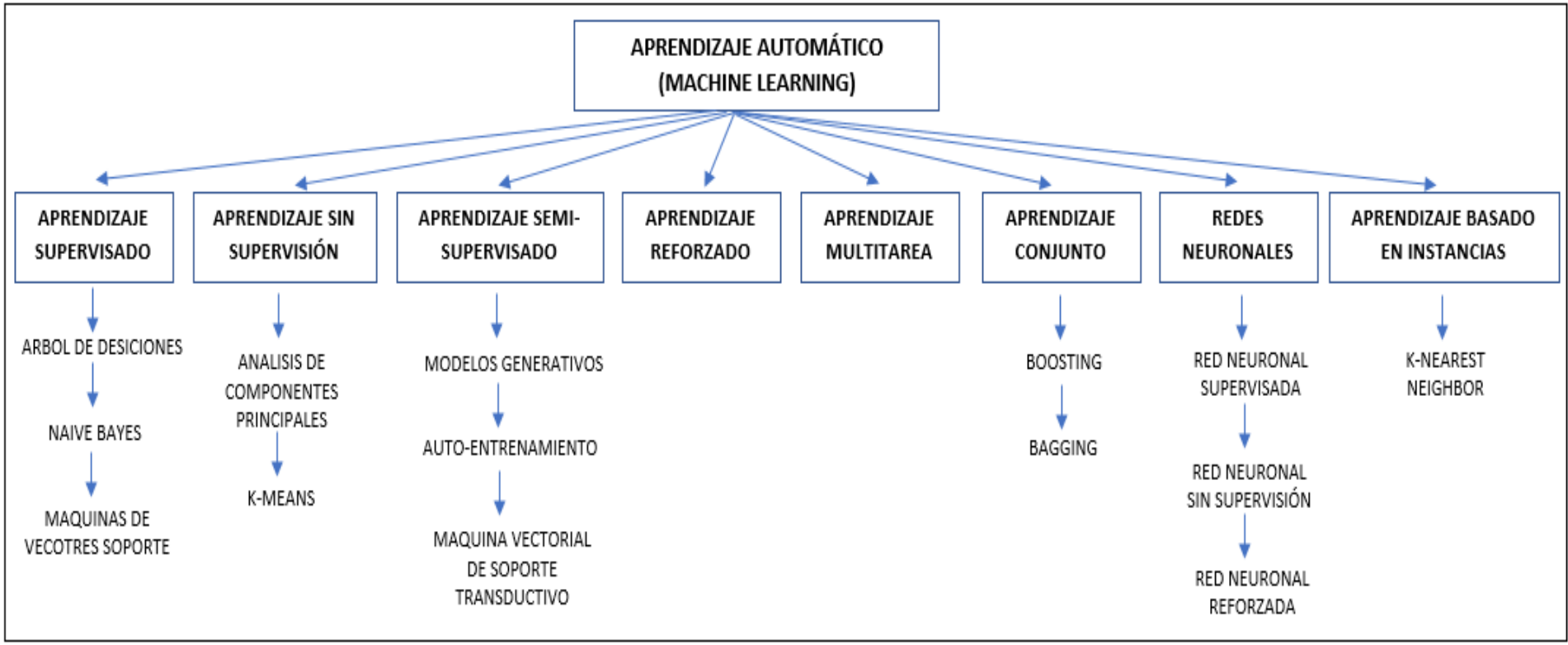
Dando paso a un sistema útil para los sistemas en tiempo real y las redes de sensores como dos exponentes de aplicaciones donde la utilización de machine learning, debe realizarse sobre dispositivos y con programación diferente a los convencionales, como las FPGA, que resultan más eficientes a la hora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

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

Tipos de aprendizaje automático



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.

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.

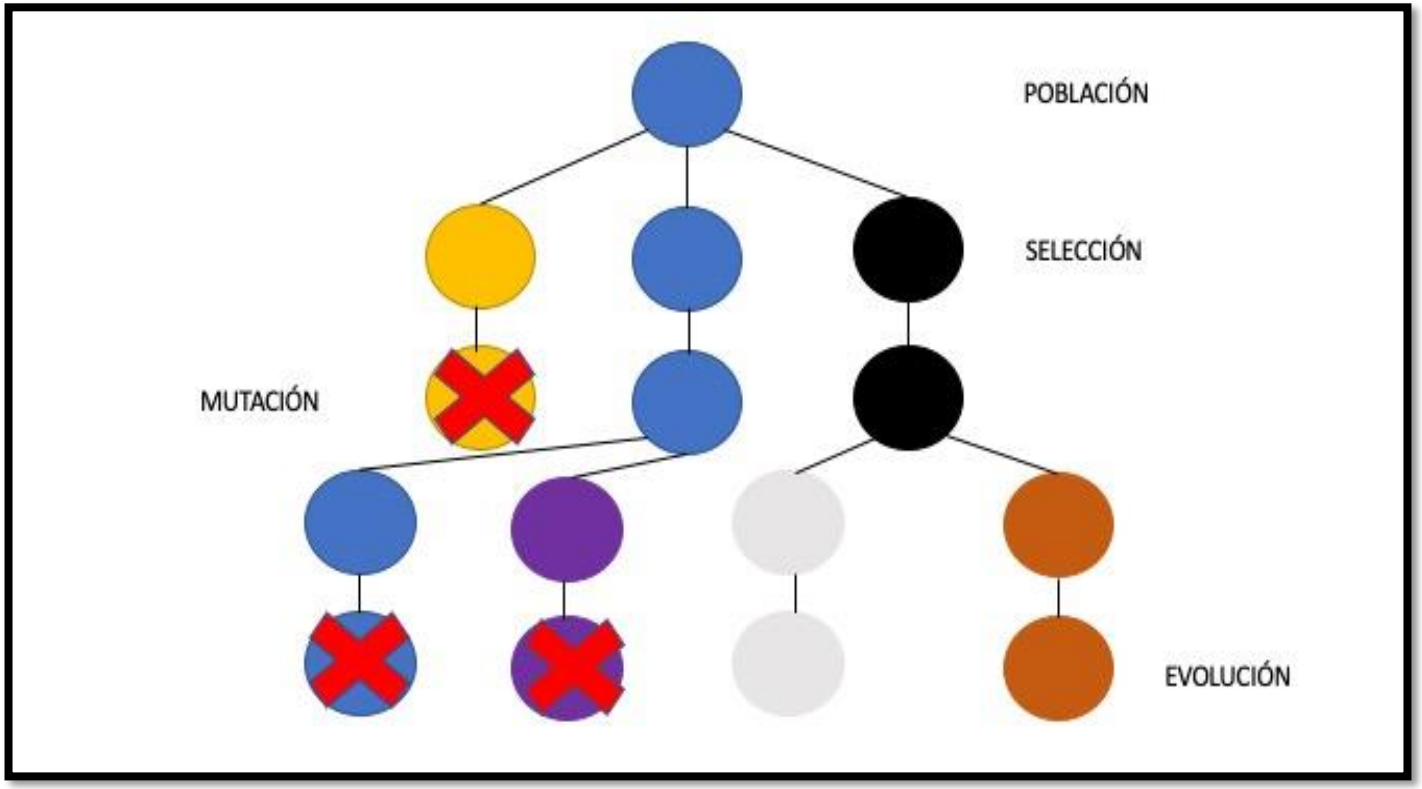
La teoría evolutiva se basa en:

- 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.

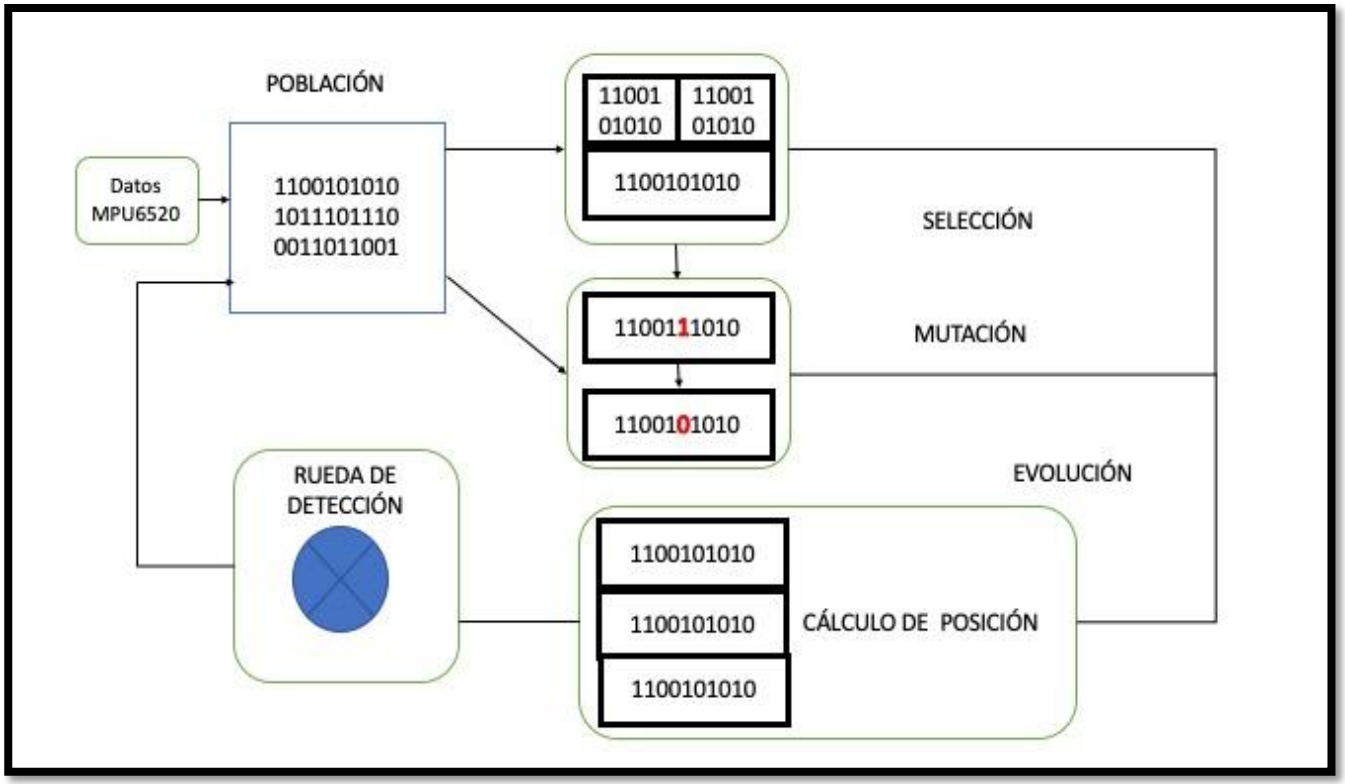
Algoritmos genéticos



Parámetros de interpretación de algoritmo genético básico

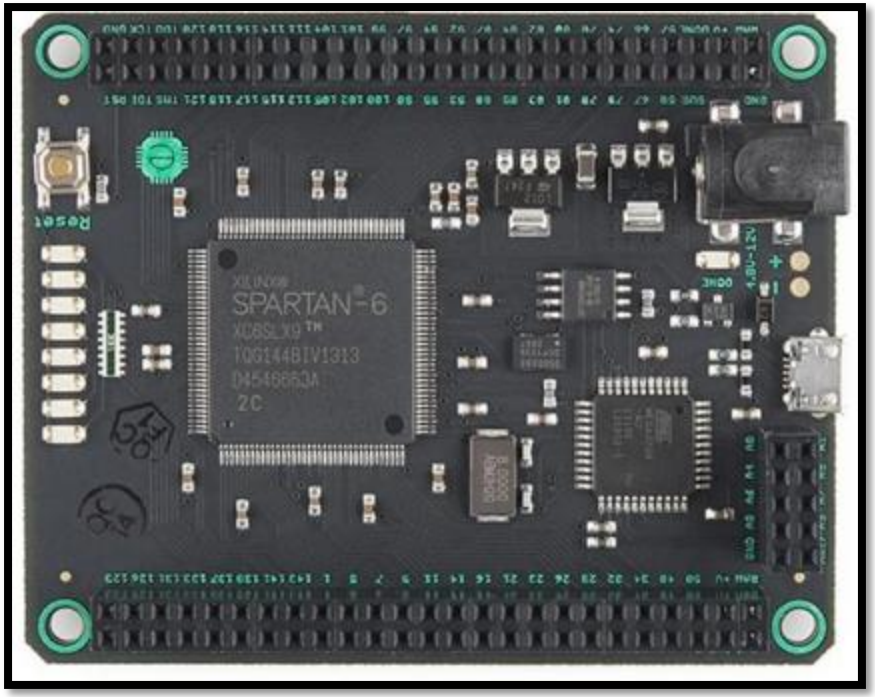
PARÁMETRO	FUNCIÓN
$\boldsymbol{\mu}$	Índice de selección
$\boldsymbol{P}$	Array de posibilidades
$\boldsymbol{x, y, z}$	Identificadores de espacio $\mathbb{R}^3$
$\boldsymbol{l}$	Contador de posiciones del vector
$\boldsymbol{fe}$	Índice de evolución
$\boldsymbol{fs}$	Índice de selección
$\boldsymbol{fm}$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

Interpretación de algoritmo genético básico

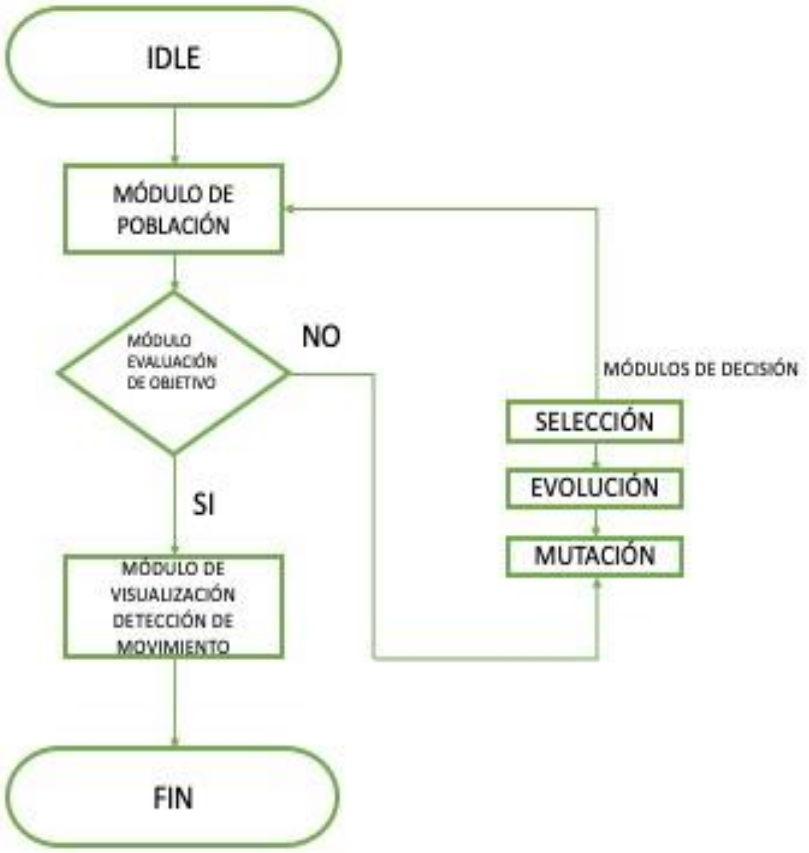


Implementación en hardware

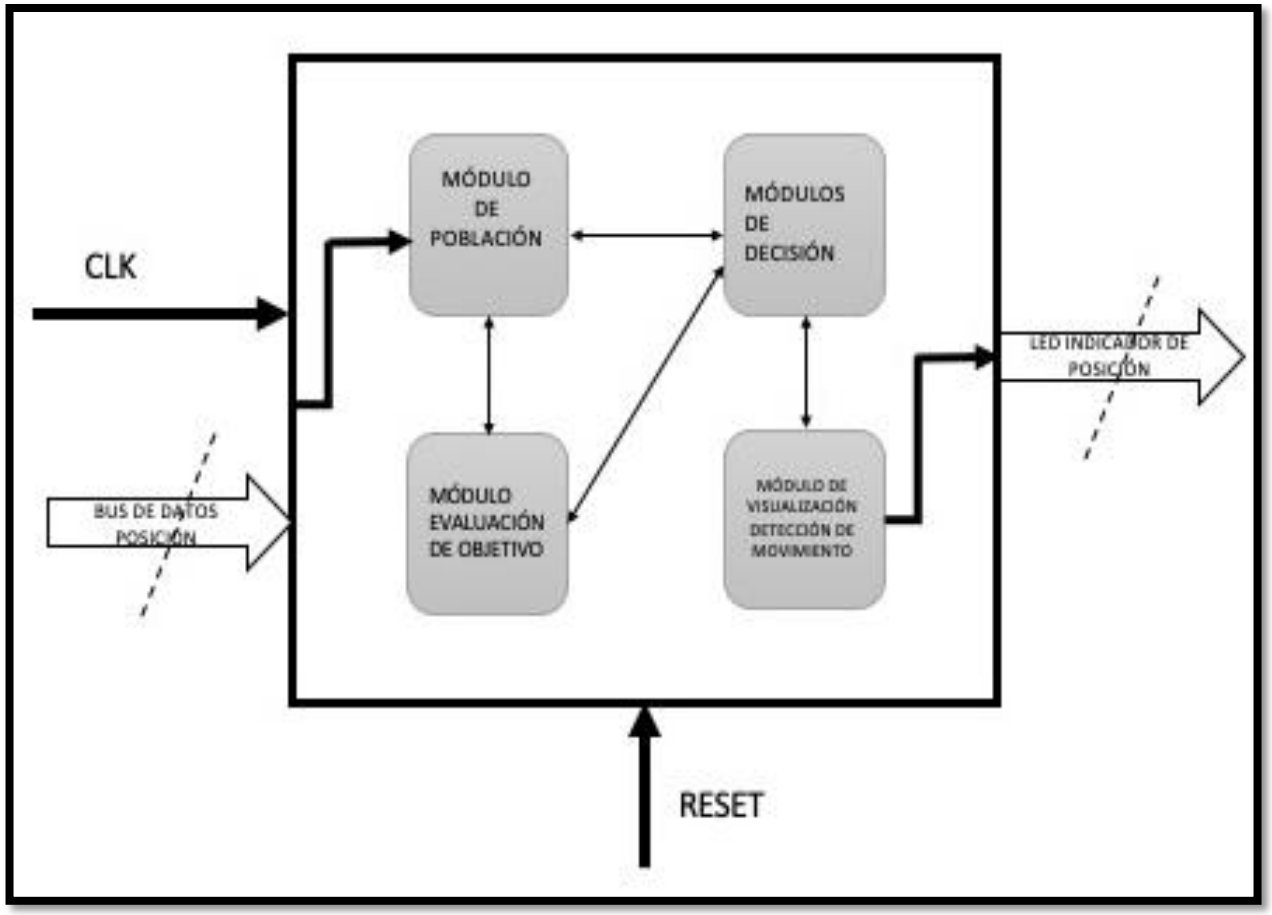
La implementación del algoritmo genético se realizó empleando una tarjeta de desarrollo MOJO V3 de ALCHINTRY, la cual contiene un FPGA Spartan 6 de XILINIX así como diversos bloques de memoria, puertos de comunicación, un indicador de posición mediante un array de leds. El software empleado para configuración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



Interpretación de algoritmo genético básico



Interpretación en bloques del algoritmo genético básico



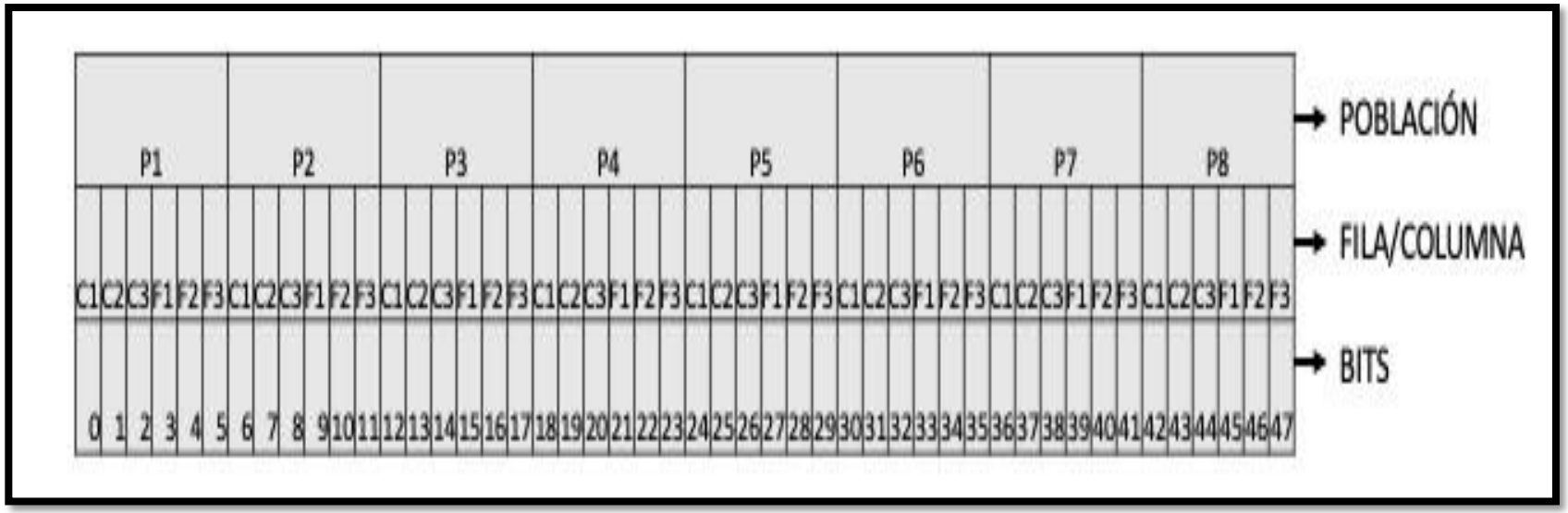
Operación del algoritmo genético en hardware

El algoritmo inicia cuando la señal de entrada (de nivel de voltaje alto), activa la funcionalidad del módulo de población

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

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



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.

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.

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

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

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

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

Caso práctico del experimento

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

$$acelhw = \frac{1}{\frac{t_{AG}}{s_{AG}} + \frac{t_{EV}}{s_{EV}}}$$

Caso práctico del experimento

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

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

Consecuentemente

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

sustituimos s_{AG} con 1,080 y sustituimos t_{AG} con $1 - t_{EV}$

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

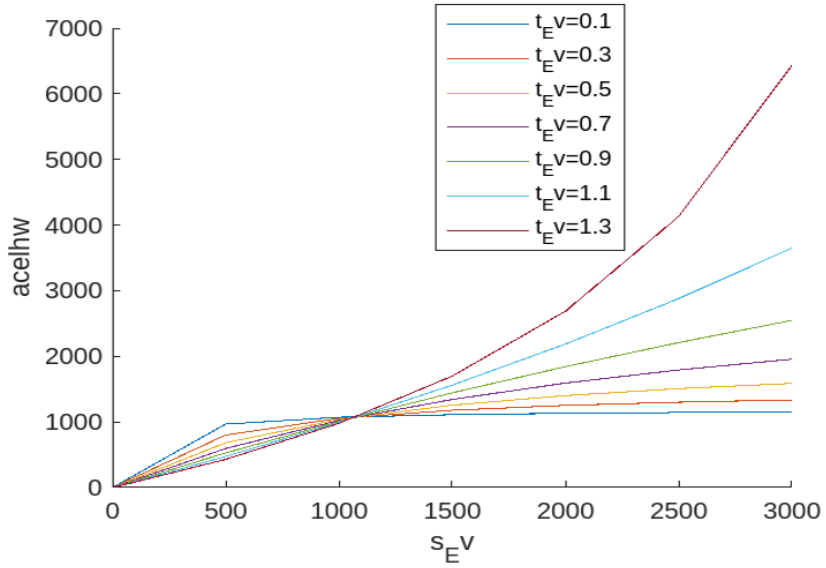
Caso práctico del experimento

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

La aceleración se muestra en la Figura 10. Cuando sEV es menor que sAG un problema cuyo tipo tEV 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 sEV es mayor que sAG, la parte dos se convierte en un cuello de botella. Como resultado, un problema cuyo tipo es tEV 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.*



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.

CONCLUSIONES

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.

RECOMENDACIONES

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.

GRACIAS POR SU ATENCIÓN