MULTIPROCESSORS- (Ruospo & Sanchez, 2021)Multiprocessor System-on-a-Chip (MPSoC)

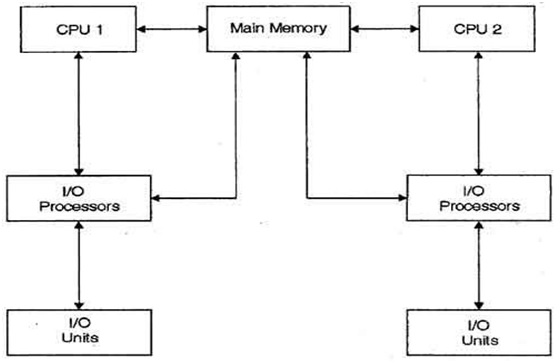
<https://www.includehelp.com/operating-systems/multiprocessor-scheduling-in-operating-system.aspx>

Sistema operativo multiprocesador

Un sistema multiprocesador consta de varios procesadores que comparten memoria. En el multiprocesador, hay más de un procesador en el sistema. La razón por la que usamos multiprocesador es que a veces la carga en el procesador es muy alta, pero no se requiere entrada y salida en otra función. Este tipo de sistema operativo es más confiable, ya que incluso si un procesador falla, el otro puede seguir funcionando. Este sistema es relativamente barato porque solo tenemos las copias del procesador pero se comparten otros dispositivos como la entrada-salida y la memoria. En el sistema multiprocesador, todos los procesadores operan bajo un único sistema operativo. La multiplicidad del procesador y la forma en que los procesadores trabajan juntos son transparentes para el otro.

En esto, el usuario no sabe en qué procesador trabaja su proceso. Un proceso se divide en varios procesos pequeños y funcionan de forma independiente en los diferentes procesadores. Un sistema puede ser multi-programado al tener múltiples programas ejecutándose al mismo tiempo y multiprocesamiento al tener más de un procesador y un físico.

En este diagrama, hay más de 1 CPU y comparten una memoria común.



ANUNCIO PUBLICITARIO

Programación multiprocesamiento

En la programación del multiprocesador, hay varias CPU que comparten la carga para que varios procesos se ejecuten simultáneamente. En general, la programación de multiprocesador es compleja en comparación con la programación de un solo procesador. En la programación multiprocesador, hay muchos procesadores y son idénticos y podemos ejecutar cualquier proceso en cualquier momento.

Las múltiples CPU del sistema están en una comunicación cercana que comparte un bus común, memoria y otros dispositivos periféricos. Entonces podemos decir que el sistema es un sistema estrechamente acoplado. Estos sistemas se utilizan cuando queremos procesar una gran cantidad de datos. Estos sistemas se utilizan principalmente en satélites, predicciones meteorológicas, etc.

El sistema de multiprocesamiento trabaja sobre el concepto de modelo de multiprocesamiento simétrico. En este sistema, cada procesador trabaja en una copia idéntica del sistema operativo y estas copias se comunican entre sí. Con la ayuda de este sistema podemos ahorrar dinero gracias a otros dispositivos como periféricos. Las fuentes de alimentación y otros dispositivos se comparten. Lo más importante es que podemos hacer más trabajo en poco tiempo. Si un sistema falla en el sistema multiprocesador, todo el sistema no se detendrá, solo la velocidad del procesador disminuirá. Todo el rendimiento del sistema de multiprocesamiento está gestionado por el sistema operativo. El sistema operativo asigna una tarea diferente a los diferentes procesadores del sistema. En el sistema de multiprocesamiento, el proceso se divide en el subproceso que se puede ejecutar de forma independiente. Este tipo de sistema permite que los subprocesos se ejecuten en más de un procesador simultáneamente. En estos sistemas, los diversos procesos en paralelo se denominan procesador paralelo. El procesamiento paralelo es la capacidad de la CPU para ejecutar varios procesos simultáneamente. En el sistema de multiprocesamiento, los recursos se comparten dinámicamente entre los distintos procesadores.

El sistema operativo multiprocesador es un tipo de sistema operativo regular que maneja muchas llamadas de sistemas al mismo tiempo, administra la memoria, proporciona administración de archivos y también los dispositivos de entrada y salida.

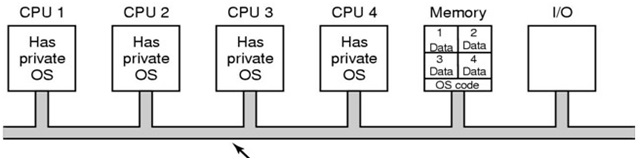
Hay algunas características adicionales que realiza el multiprocesador:

* Sincronización de procesos
* Administracion de recursos
* Planificación

Hay varias organizaciones de sistema operativo multiprocesador:

1. Cada CPU tiene su propio sistema operativo

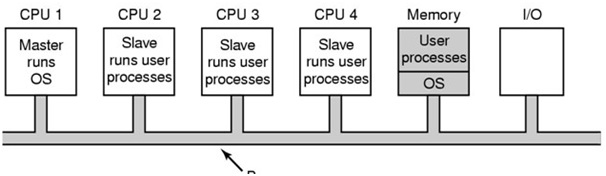
En este tipo de organización, hay muchas unidades centrales de procesamiento en el sistema y cada CPU tiene su propio sistema operativo privado y la memoria se comparte entre todos los procesadores y el sistema de entrada y salida también se comparte. Todo el sistema está conectado por un solo bus.



ANUNCIO PUBLICITARIO

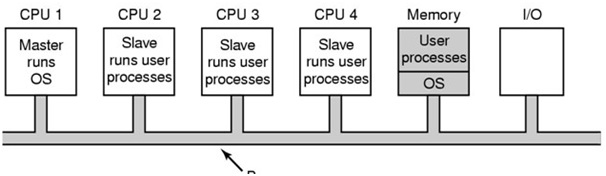
2. Multiprocesador maestro esclavo

En este tipo de modelo de multiprocesador, hay una única estructura de datos que realiza un seguimiento de los procesos listos. En este modelo, una unidad central de procesamiento funciona como maestra y otra unidad central de procesamiento funciona como esclava. En esto, todos los procesadores son manejados por un solo procesador que se llama servidor maestro. El servidor maestro ejecuta el proceso del sistema operativo y el servidor esclavo ejecuta los procesos del usuario. La memoria y los dispositivos de entrada-salida se comparten entre todos los procesadores y todos los procesadores están conectados a un bus común. Este sistema es simple y reduce el intercambio de datos, por lo que este sistema se llama **multiprocesamiento asimétrico** .



3. Multiprocesador simétrico

Los multiprocesadores simétricos (SMP) es el tercer modelo. En este modelo, hay una copia del sistema operativo en la memoria, pero cualquier unidad central de procesamiento puede ejecutarla. Ahora, cuando se realiza una llamada al sistema, la unidad central de procesamiento en la que se realizó la llamada al sistema captura el kernel y luego procesa esa llamada al sistema. Este modelo equilibra los procesos y la memoria de forma dinámica. Este enfoque utiliza multiprocesamiento simétrico donde cada procesador se autoprograma. La programación prosigue al hacer que el programador de cada procesador examine la cola lista y seleccione un proceso para ejecutar. En este sistema, es posible que todo el proceso esté en una cola lista común o que cada procesador tenga su propia cola privada para el proceso listo.

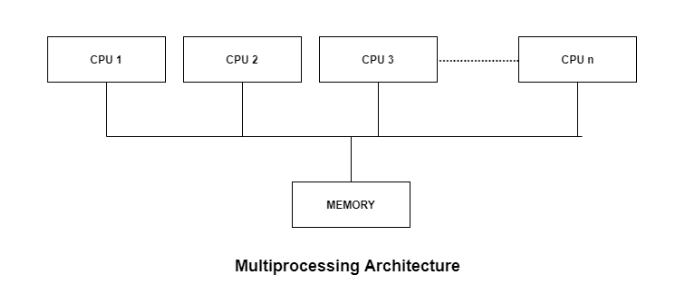


**Hay principalmente tres fuentes de controversia que se pueden encontrar en un sistema operativo multiprocesador.**

* **Sistema de bloqueo:** Como sabemos que los recursos se comparten en el sistema multiprocesador, existe la necesidad de proteger estos recursos para un acceso seguro entre los múltiples procesadores. El propósito principal del esquema de bloqueo es serializar el acceso a los recursos por parte de múltiples procesadores.
* **Datos compartidos:** Cuando el procesador múltiple accede a los mismos datos al mismo tiempo, puede haber una posibilidad de inconsistencia de datos, por lo que para protegerlos tenemos que usar algunos protocolos o esquemas de bloqueo.
* **Coherencia de la caché:** Son los datos de recursos compartidos que se almacenan en las múltiples cachés locales. Suponga que hay dos clientes que tienen una copia de memoria en caché y un cliente cambia el bloque de memoria y el otro cliente podría quedarse con una caché no válida sin notificación del cambio, por lo que este tipo de conflicto se puede resolver manteniendo una vista coherente de los datos.

…………………………………………

<https://www.tutorialspoint.com/Multiprocessor-Systems>

Most computer systems are single processor systems i.e they only have one processor. However, multiprocessor or parallel systems are increasing in importance nowadays. These systems have multiple processors working in parallel that share the computer clock, memory, bus, peripheral devices etc. An image demonstrating the multiprocessor architecture is − 

## Types of Multiprocessors

There are mainly two types of multiprocessors i.e. symmetric and asymmetric multiprocessors. Details about them are as follows −

### Symmetric Multiprocessors

In these types of systems, each processor contains a similar copy of the operating system and they all communicate with each other. All the processors are in a peer to peer relationship i.e. no master - slave relationship exists between them.

An example of the symmetric multiprocessing system is the Encore version of Unix for the Multimax Computer.

### Asymmetric Multiprocessors

In asymmetric systems, each processor is given a predefined task. There is a master processor that gives instruction to all the other processors. Asymmetric multiprocessor system contains a master slave relationship.

Asymmetric multiprocessor was the only type of multiprocessor available before symmetric multiprocessors were created. Now also, this is the cheaper option.

## Advantages of Multiprocessor Systems

There are multiple advantages to multiprocessor systems. Some of these are −

**More reliable Systems**

In a multiprocessor system, even if one processor fails, the system will not halt. This ability to continue working despite hardware failure is known as graceful degradation. For example: If there are 5 processors in a multiprocessor system and one of them fails, then also 4 processors are still working. So the system only becomes slower and does not ground to a halt.

**Enhanced Throughput**

If multiple processors are working in tandem, then the throughput of the system increases i.e. number of processes getting executed per unit of time increase. If there are N processors then the throughput increases by an amount just under N.

**More Economic Systems**

Multiprocessor systems are cheaper than single processor systems in the long run because they share the data storage, peripheral devices, power supplies etc. If there are multiple processes that share data, it is better to schedule them on multiprocessor systems with shared data than have different computer systems with multiple copies of the data.

## Disadvantages of Multiprocessor Systems

There are some disadvantages as well to multiprocessor systems. Some of these are:

**Increased Expense**

Even though multiprocessor systems are cheaper in the long run than using multiple computer systems, still they are quite expensive. It is much cheaper to buy a simple single processor system than a multiprocessor system.

**Complicated Operating System Required**

There are multiple processors in a multiprocessor system that share peripherals, memory etc. So, it is much more complicated to schedule processes and impart resources to processes.than in single processor systems. Hence, a more complex and complicated operating system is required in multiprocessor systems.

**Large Main Memory Required**

All the processors in the multiprocessor system share the memory. So a much larger pool of memory is required as compared to single processor systems.

…………………………………………

**MULTIPROCESSOR SYSTEM ON A CHIP**

Un sistema multiprocesador en un chip (MPSoC) es un sistema en un chip (SoC) que incluye múltiples microprocesadores. Como tal, es un sistema de múltiples núcleos en un chip.

Los MPSoC suelen estar destinados a aplicaciones integradas. Es utilizado por plataformas que contienen múltiples elementos de procesamiento, generalmente heterogéneos, con funcionalidades específicas que reflejan la necesidad del dominio de aplicación esperado, una jerarquía de memoria y componentes de E / S. Todos estos componentes están vinculados entre sí mediante una interconexión en chip, como buses y redes en chip (NoC). Estas arquitecturas satisfacen las necesidades de rendimiento de aplicaciones multimedia, arquitecturas de telecomunicaciones, seguridad de red y otros dominios de aplicaciones, al tiempo que limitan el consumo de energía mediante el uso de arquitectura y elementos de procesamiento especializados.

**On the Reliability Assessment of Artificial Neural Networks**

**Running on AI-Oriented MPSoCs**

**Annachiara Ruospo \* and Ernesto Sanchez**

\_\_\_\_\_\_\_\_\_\_

**\_\_\_\_\_\_\_**

**Citation:** Ruospo, A.; Sanchez, E. On

the Reliability Assessment of

Artificial Neural Networks Running

on AI-Oriented MPSoCs. Appl. Sci.

**2021**, 11, 6455. https://doi.org/

10.3390/app11146455

Academic Editor: Arcangelo

Castiglione

Received: 11 June 2021

Accepted: 9 July 2021

Published: 13 July 2021

**Publisher’s Note:** MDPI stays neutral

with regard to jurisdictional claims in

published maps and institutional affiliations.

**Copyright:** ©

**Resumen:** Hoy en día, el uso de dispositivos electrónicos que ejecutan aplicaciones basadas en redes neuronales artificiales (ANN) se está extendiendo en nuestra vida cotidiana. Debido a sus excepcionales capacidades computacionales, las ANN se han convertido también en soluciones atractivas para sistemas críticos para la seguridad. Con frecuencia se los considera intrínsecamente robustos y tolerantes a fallas por ser modelos de computación redundantes e inspirados en el cerebro. Sin embargo, cuando las ANN se implementan en dispositivos de hardware con recursos limitados, las fallas físicas únicas pueden comprometer la actividad de múltiples neuronas. Por lo tanto, es crucial evaluar la confiabilidad de todo el sistema de computación neuronal, incluidos los componentes de software y hardware. Este artículo aborda sistemáticamente los problemas de confiabilidad de las ANN que se ejecutan en sistemas multiprocesador en chips (MPSoC). Presenta una metodología para asignar puntajes de resiliencia a neuronas individuales y, en base a eso, programar la carga de trabajo de una ANN en el MPSoC objetivo para que las neuronas críticas se distribuyan ordenadamente entre los elementos de procesamiento disponibles. Esta metodología orientada a la confiabilidad explota un solucionador de programación lineal de enteros para encontrar la solución óptima. Se dan resultados experimentales para tres redes neuronales convolucionales diferentes entrenadas en MNIST, SVHN y CIFAR-10. Realizamos una evaluación integral de un MPSoC RISC-V basado en inteligencia artificial de código abierto. Los resultados muestran las mejoras de confiabilidad de la metodología propuesta frente a la programación tradicional.

**Keywords:** artificial neural network; reliability; fault tolerance

…..

Nevertheless, we advocate that the theory claiming the intrinsic ANNs fault tolerance may hold true only in two cases: if neural networks are merely viewed as a software and a mathematical abstraction; if there is a one-to-one correspondence between the neurons and the hardware processing elements (PEs) on which the ANN model runs.

……

To meet all these requirements, multiprocessor systems-on-a-chip (MPSoCs) currently represent the best option for running AI-based applications [12,13]. They are heterogeneous SoCs made of multiple CPUs and/or multiple PEs along with other hardware subsystems, such as specific hardware accelerators [14–16].

Motivated by the above-mentioned considerations, the intent of the article is manyfold. The first goal is to tackle the reliability aspect of neural networks with the aim of demonstrating that they are not generally intrinsically resilient; their reliability must be evaluated with respect to the intended hardware implementation. In this regard, the second goal of the article is to propose a methodology to improve the reliability of NCSs based on resource-constrained MPSoCs. The main contributions of this paper are listed in the following:

We present a methodology to identify the most critical neurons of a neural network by assigning resilience values to each of them. The method bases on two levels of analysis: first, the neuron is viewed as an element of each output class (class-oriented analysis); second, the same is interpreted as belonging to the entire neural network (network-oriented analysis). The method can be efficiently applied to neural networks with any layers and any typologies. The methodology is validated by means of software fault injection (FI) campaigns, using three different convolutional neural networks (CNNs) trained on three different data sets: MNIST, SVHN, and CIFAR-10.

• Based on the above criticality analysis, we describe an approach to evenly distribute critical neurons among the available PEs of the MPSoC to improve the reliability of the NCS. It exploits integer linear programming (ILP) to find the optimal and deterministic solution to map ANNs elaborations onto the target hardware architecture. To prove the effectiveness of this reliability-oriented approach, we carried out FI campaigns at the register transfer level (RTL) on an open-source RISC-V MPSoC for AI at the edge, i.e., the GAP-8 architecture [16]. Specifically, to understand the vulnerability of the MPSoC-based NCS to random hardware faults, permanent faults are addressed in this work. Recent works have highlighted that permanent faults in DNN accelerators have a major impact on DNN accuracy with respect to, for instance, temporary faults (soft errors) [18].

**3. Proposed Approach**

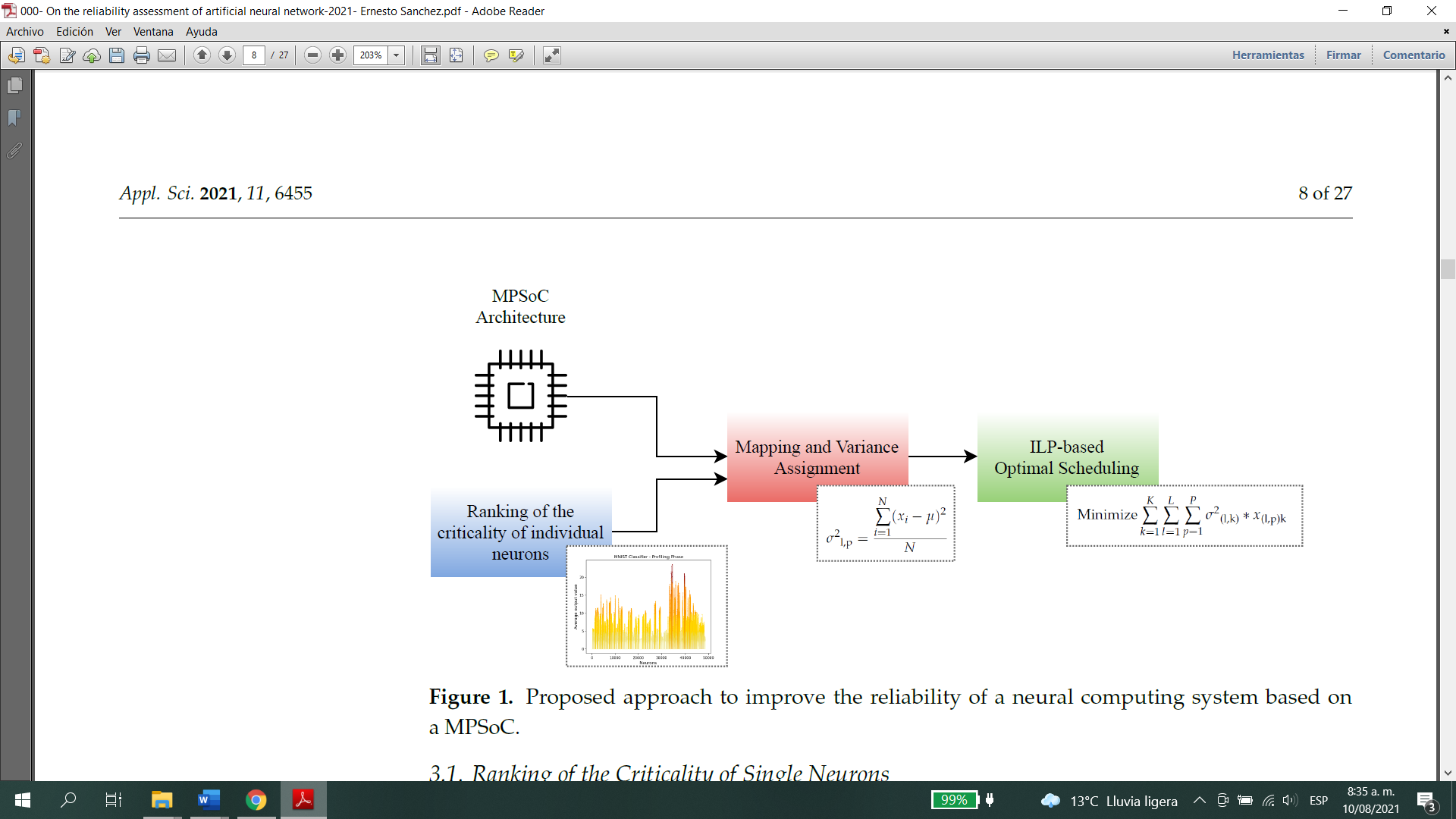
The proposed methodology is based on the identification of the most critical neurons inside the ANN to then determine the best scheduling of the ANN application workload in the targeted MPSoC. We assume that the ANN is ready to be deployed on the intended hardware architecture and any modifications of the ANN application are not required. Indeed, only the pretrained ANN application and the available hardware resources are considered. As experimentally demonstrated, the proposed technique is capable of increasing the reliability of the NCS. In this work, we present a methodology that is built on the following three steps:

1. **Ranking of the criticality of single neurons**: Resilience scores are assigned to individual neurons of the ANN.

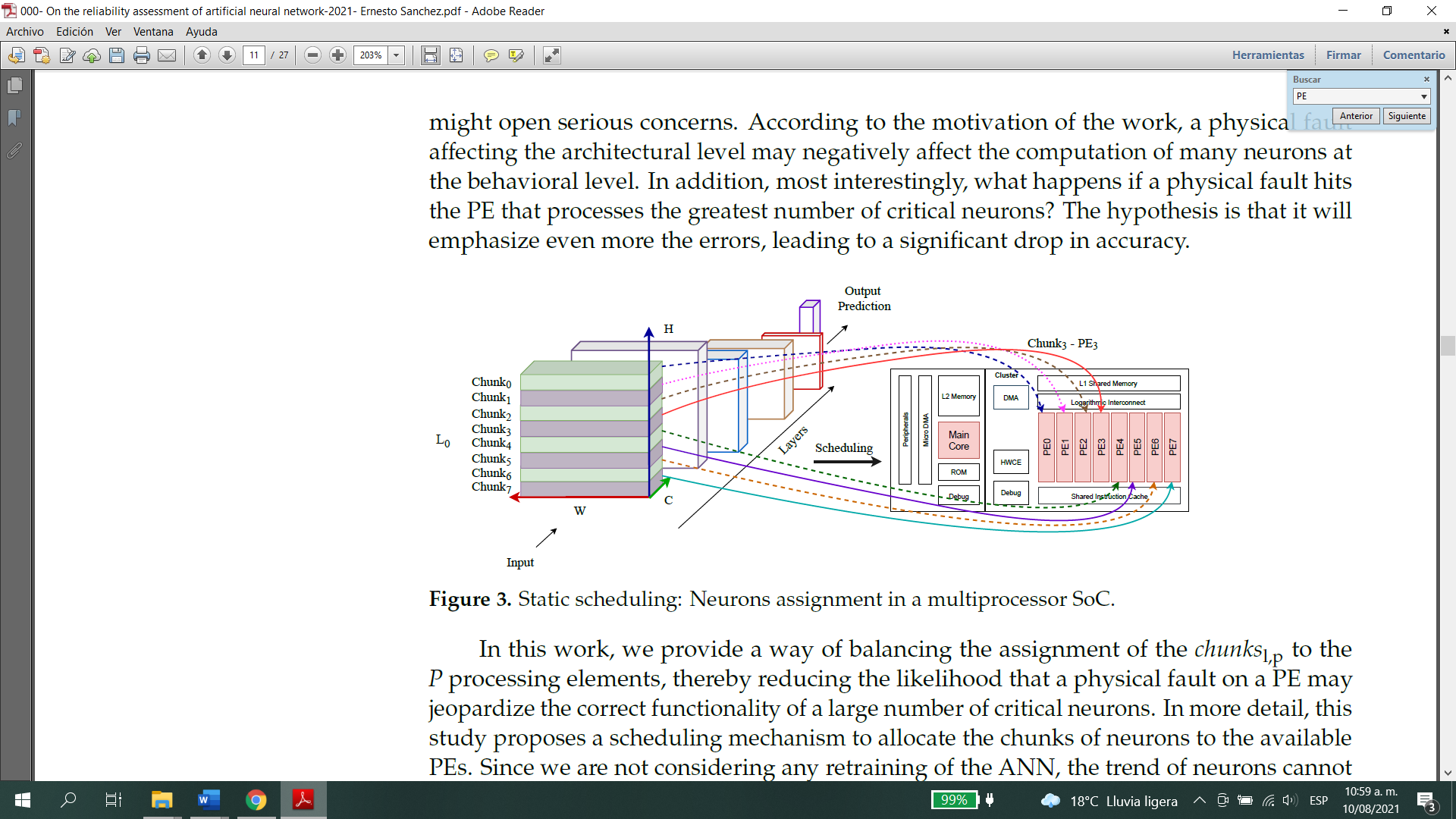
2. **Mapping and variance assignment:** Based on the previous phase and on the available PEs of the target AI-oriented MPSoC, a value is given to each chunk of neurons assigned to a single PE. We adopt a mathematical metric as a decision-making parameter—the variance. This value indicates the criticality of the chunk; in other words, the amount of critical neurons in that chunk that are assigned to a PE.

3. **ILP-based optimal scheduling**: By leveraging on the chunks variance, an ILP solver is set up to obtain the optimal reliability-oriented scheduling for mapping ANN inferences on a specific hardware device.

These three phases are shown in Figure 1 and detailed in the following.



Traditionally, to improve parallelism, AI-oriented MPSoCs distribute the ANN workload, exploiting the SIMD paradigm. As shown in Figure 3, the neurons N = {0, . . . , n} are neatly distributed among all the P = {0, . . . , p} PEs, and it is known exactly which neurons that a PE handles when launching the inference of a L-layer neural network, where L = {0, . . . , l}. Thus, it is possible to split the total amount of neurons in well-defined chunks, consisting of fixed groups of neurons assigned, at each layer l, to a specific PE p. Nevertheless, the amount of critical neurons belonging to each chunkl,p is not equally distributed among the P computing resources.



Conclusions

Este documento proporciona una metodología para mejorar la confiabilidad de un sistema de computación neuronal que se ejecuta en un dispositivo de múltiples núcleos. A través del artículo, se demostró que es posible identificar las neuronas más críticas de una red neuronal y, en base a esto, determinar una programación óptima para un MPSoC orientado a IA. Siguiendo la metodología propuesta, se demostró experimentalmente que no todas las neuronas ANN juegan el mismo papel en la tarea final. Es justo decir que las redes neuronales están equipadas con más neuronas de las necesarias, pero qué neuronas eliminar es el foco del análisis orientado a clases. Inmediatamente después de este estudio, presentamos una técnica para identificar las neuronas más críticas de la ANN y las clasificamos según su criticidad. Los resultados experimentales muestran que nuestra clasificación final es más efectiva que las basadas únicamente en un análisis orientado a la red, ya que también consideramos la criticidad de las neuronas con respecto a las clases de salida. Basándose en este análisis, el documento introdujo un mecanismo basado en programación lineal de enteros, que tiene en cuenta la métrica de varianza de porciones de neuronas. El objetivo era distribuir uniformemente las neuronas críticas a los elementos de procesamiento disponibles. Los resultados de una nueva campaña de inyección en RTL proporcionan evidencia de que la programación propuesta puede enmascarar los efectos de más fallas y predecir menos predicciones erróneas. Se obtuvo una reducción del 24,74% de las predicciones erróneas y una mejora del 97,80% y del 59,53% de las fallas enmascaradas. Vale decir que, sin recurrir a ninguna técnica basada en redundancia (ya sea software o hardware), se puede mejorar la confiabilidad de un NCS y su tolerancia a fallas.

El trabajo futuro extenderá este análisis a RNA más profundas y diferentes conjuntos de datos. El lector debe notar que la adopción de los conjuntos de datos MNIST, SVHN y CIFAR-10 fue consistente con el mundo ASIC considerado de bajo consumo de energía y recursos limitados. En el futuro, aprovecharemos ANN más profundos y conjuntos de datos más complejos, trasladando el objetivo a GPU y arquitecturas de alto rendimiento o chips de red neuronal de hardware ad hoc. Claramente, con redes neuronales más complejas, necesitamos estudiar la viabilidad de la distribución uniforme y, si esto no se puede satisfacer, evaluaremos y propondremos diferentes estrategias que se adapten bien a la complejidad de la aplicación. Para concluir, en el futuro, abordaremos también otros modelos de fallas, como los errores transitorios.

Xxxxxx

En (xxx) se presenta una estudio de una ANN implementada sobre un dispositivo de múltiples núcleos MPSoC

xxxxxx

ANDROMEDA: An FPGA Based RISC-V MPSoC

Exploration Framework

Resumen: Con la creciente demanda de productos electrónicos de consumo, los requisitos computacionales aumentan exponencialmente. Debido a las necesidades computacionales de las aplicaciones, los arquitectos de computadoras están tratando de empaquetar tantos núcleos como sea posible en un solo dado para la ejecución acelerada de los códigos del programa de la aplicación. En un sistema en chip multiprocesador (MPSoC), lograr un equilibrio entre el número de núcleos, subsistemas de memoria y parámetros de red en chip es esencial para lograr el rendimiento deseado. En este artículo, presentamos ANDROMEDA, un marco basado en RISC-V que nos permite explorar las diferentes configuraciones de un MPSoC y observar las penalizaciones y ganancias de rendimiento. Emulamos las diversas configuraciones de MPSoC en la plataforma Synopsys HAPS-80D Dual FPGA. Al utilizar STREAM, la multiplicación de matrices y las simulaciones de N cuerpos como puntos de referencia, demostramos la eficacia de nuestro marco para identificar rápidamente los parámetros correctos para la ejecución eficiente de estos puntos de referencia.

Index Terms—design space exploration, multiprocessor systemon-chip, RISC-V, performance tuning

Acceleration of deep convolutional neural networks on multiprocessor system-on-chip

Efficient Memory Organization for DNN Hardware Accelerator Implementation on PSoC

A Deep Learning-Based FPGA Function Block Detection Method with Bitstream to Image Transformation

Ruospo, A., & Sanchez, E. (2021). On the Reliability Assessment of Artificial Neural Networks Running on AI-Oriented MPSoCs. *Applied Sciences*, *11*(14), 6455. https://doi.org/10.3390/app11146455