UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA
INGENIERIA ELÉCTRICA – SISTEMAS ELECTRÓNICOS

DISEÑO DE UNA ARQUITECTURA DE RED NEURONAL EN FPGA PARA LA NAVEGACIÓN VISUAL EN MARTE

TESIS
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN INGENIERÍA

PRESENTA:
DIEGO FRANCISCO MARTÍNEZ VALDÉS

TUTOR:
SAÚL DE LA ROSA NIEVES
FACULTAD DE INGENIERÍA, UNAM

CDMX, JUNIO 2021

Jurado Asignado:

Presidente: Dra. Tetyana Mykolaivna Baydyk

Secretario: Dr. Jorge Rodríguez Cuevas

1er Vocal: Dr. Saúl de la Rosa Nieves

2do Vocal: Dr. Ernst Mikhailovich Kussul

3es Vocal: Dr. Juan Mario Peña Cabrera

LUGAR DONDE SE REALIZÓ LA TESIS:
Laboraratorio de Instrumentación Electrónica de Sistemas Espaciales, Facultad de ingeniería, UNAM, México

Tutor de Tesis:

Dr. Saúl de la Rosa Nieves

Firma

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ACRÓNIMOS

ALU Arithmetic Logic Unit
ANN Artificial Neural Networks
ASIC Application-Specific Integrated Circuit
IDE Integrated Development Environment
ITM Imagenes del Terreno Marciano
BP Backpropagation
CC Corriente Continua
CCD Charge Coupled Device
CLA Carry-lookahead adder
CLB Configurable Logic Block
CMOS Complementary Metal-Oxide Semiconductor
CPS Connections Per Second
CPU Central Processing Unit
CUPS Connection Updates Per Second
CVAC Computer Vision Accelerator Card
DRAM Dynamic Random Access Memory
DSP Digital Signal Processors
EEPROM Electrically Erasable Programmable Read-Only Memory
EUA Estados Unidos de America
FA Full Adder
FG Frame Grabber
FPGA Field Programmable Gate Arrays
FPNA Field Programmable Neural Arrays
FPNN Field Programmable Neural Networks
GLC Generador Lineal Congruencial
GPU Graphics Processing Unit
GPS Global Positioning System
GUI Graphical User Interface
HDL Hardware Description Language
HDR High Dynamic Range
HMB full name
IA Inteligencia Artificial

IOB input/output block
JPL Jet Propulsion Laboratory
LAB Logic Array Block
LIRA Limited Receptive Area
LUT Look Up Tables
MAC Multiplicación-acumulación
MAHLI Mars Hand Lens Imager
MER Mars Exploration Rover
MIMD Multiple Instruction, Multiple Data
MSL Mars Science Laboratory
NASA National Aeronautics and Space Administration
NIR Near Infrared
OTP One Time Programmable
PCNC Permutation Coding Neural Classifier
PDS Planetary Data System
PIA Planetary Image Atlas
RCA Ripple-carry adder
RCE Rover Compute Element
REM Rover Electronics Module
RSC Random Subspace Classifier
RTC Random Threshold Classifier
RTL Register Transfer Logic
SELENE Selenological and Engineering Explorer
SIMD Single instruction, multiple data
SLAM Simultaneous Localization and Mapping
SMA Shape-memory alloy
SNC Sistema nervioso central
SNP Sistema nervioso periferico
SNN Spiking Neural Network
TDR Tracking and Data Relay
URSS Unión de Repúblicas Socialistas Soviéticas
VME VERSA Module Eurocard
WEB Warm Electronic Box

INTRODUCCIÓN

La exploración robótica de las superficies planetarias y el espacio profundo exige que las tecnologías utilizadas funcionen, casi en su totalidad, de manera autónoma. La teleoperación no es una opción práctica para misiones más allá de la distancia a la que se encuentra la Luna debido al tiempo que tarda el enlace bidireccional, se necesitan varios minutos entre el instante en que la cámara de un móvil detecta un obstáculo en su camino y cuando llegan los comandos de corrección de rumbo desde la tierra. La dependencia de la teleoperación en diferentes tareas secundarias durante la navegación del robot ocasiona que el funcionamiento sea lento y puede derivar en problemas de incertidumbre e inseguridad para el robot en el terreno que transita.

Los rover normalmente necesitan moverse constantemente a nuevos puntos destino con el fin de llevar a cabo nuevas tareas de exploración. Por lo tanto, en futuras misiones espaciales, los algoritmos de navegación autónoma son esenciales para que los robots puedan evadir áreas de riesgo (tales como cráteres, altas montañas y rocas) y llegar a los puntos destino de manera precisa y eficiente.

Para que los robots de exploración tengan la capacidad de navegar por ambientes abiertos, desconocidos e inesperados es necesario que estos no solo tengan la capacidad de determinar la posición en la se encuentran, sino también reconocer los obstáculos y otros elementos de su al rededor de forma rápida y precisa. Si bien es cierto que en la actualidad existen muchas tecnológicas diferentes para realizar técnicas de localización y mapeo simultáneo (SLAM, por sus siglas en inglés), no todas son recomendables, en la actualidad, para aplicaciones de exploración espacial, debido a las restricciones en cuanto a energía y capacidad de memoria de los sistemas. Los algoritmos de aprendizaje automático son cada vez más utilizados, su buen desempeño combinado con las tecnologías electrónicas reconfigurables y de gran velocidad permiten realizar sistemas confiables que realicen tareas en tiempo real.

En este trabajo de tesis se presenta el diseño de la implementación en FPGA del clasificador neuronal LIRA-escala de grises para el reconocimiento de texturas del terreno marciano, se describe la teoría de las redes neuronales, su lugar dentro del aprendizaje automático y las metodológicas para su utilización en sistemas de descripción de hardware. El trabajo no solo muestra los resultados del diseño, también sirve como guía y metodología para la implementación de redes neuronales en FPGA. Se hace especial énfasis su aplicación (Marte y los robots exploradores), los antecedentes y la motivación de trabajar en este tema, ya que es de gran interés para el laboratorio generar una buena documentación que sirva como base para seguir desarrollando proyectos sobre esta línea de trabajo.

0.1 Justificación

Para satisfacer la necesidad de tener robots que puedan realizar el proceso de navegación de manera autónoma en ambientes abiertos y desconocidos, es necesario desarrollar sistemas que doten al robot de la cualidad de reconocer su entorno e identificar el mundo que lo rodea, si el robot puede detectar, reaccionar y realizar acciones en respuesta al ambiente en el que se encuentra de manera autónoma, se puede reservar el proceso de la teleoperación, el cual requiere mucho tiempo, para situaciones particulares y que necesariamente requieran la intervención de las decisiones humanas.

Así mismo se deben diseñar sistemas que tengan la capacidad de servir para más de una tarea específica, que puedan en cierta medida adaptarse a las situaciones que el robot enfrenta en cada instante y recaben la mayor información útil posible. Es necesario mejorar cada vez más las capacidades de sistemas autónomos así como su confiabilidad y eficiencia para que trabajen en ambientes inesperados y aislados en comunicación con el exterior, tales como cuevas, cavernas o cráteres, dentro o fuera del paleta tierra, las cuales son de gran interés científico.

La Terramecánica desempeña un papel fundamental en las áreas de vehículos terrestres y robots móviles terrestres, ya que comprender y estima las variables que influyen en la interacción vehículo-terreno, esto puede significar el éxito o el fracaso de una misión, por lo que es de gran interés aplicar algoritmos de vanguardia para clasificación del terreno que en el que navega un robot autónomo.

0.2 Objetivo

- Diseñar la implementación en hardware de un sistema reconocimiento de patrones en base a redes neuronales, con el fin de obtener y procesar información visual que sea de utilidad para la navegación autónoma de un robot explorador de superficie marciana.

0.2.1 *Objetivos particulares*

- Entrenar y probar el algoritmo de reconocimiento con las imágenes de la base de datos Planetary Data System de la NASA.
- Obtener el valor óptimo de los parámetros de la red neuronal para la tarea de aplicación de reconocimiento de terreno.
- Realizar el diseño de la implementación de una arquitectura de red neuronal en FPGA.

Parte I

ANTECEDENTES

Si he llegado a ver más lejos, es porque estoy sentado sobre los hombros de gigantes

— Isaac Newton

1

EL PLANETA ROJO

El espacio: la última frontera, nada impulsa igual al espíritu humano como la posibilidad de explorar mundos desconocidos, descubrir nuevas formas de vida y conocer nuevas civilizaciones. Este primer capítulo está dedicado a uno de los planetas con mayor interés científico de la actualidad, el cual los antepasados incluso reconocían con un dios. Aquí se describe brevemente sus principales características así como la importancia e ilusión en torno a ser explorado.

Desde hace miles de años, el ser humano observaba el cielo y se daba cuenta que muchas estrellas brillantes cambiaban de lugar noche tras noche. Los antiguos griegos la llamaban a estas estrellas errantes *planetas*.[1] Uno de estos planetas en particular, posee un color anaranjado rojizo que algunas civilizaciones antiguas lo asociaron con el color de la sangre. El planeta Marte, quien recibe su nombre en honor al dios de la guerra de la mitología romana (conocido como Ares en la mitología griega) y que de ahí se derivan, a su vez, los nombres del mes de marzo, el día de la semana martes y los nombres Martín y Martínez. El conocimiento de la existencia de Marte data de las primeras civilizaciones, y desde entonces ha invadido la curiosidad y la imaginación de los seres humanos, tanto por su cercanía a la tierra como por su particular color, el cual se atribuye al oxido de hierro predominante en su regolito.[2] Marte es el cuarto y último planeta rocoso de nuestro sistema solar, también es el planeta que más ha estimulado la imaginación de las personas, en la actualidad es el planeta más interesante para realizar las primeras misiones de exploración robótica. Pero, ¿por qué tanta fascinación por este planeta en particular?, quizá la respuesta nos las pueda dar uno de los mayores divulgadores científicos de la historia:

> «£Por qué marcianos?, £por qué tantas especulaciones vehementes y tantas fantasías desbocadas sobre los marcianos y no por ejemplo, sobre los saturnianos o plutonianos?. Pues porque Marte parece, a primera vista, muy semejante a la tierra. Es el planeta más próximo con una superficie visible; hay casquetes polares de hielo, blancas nubes a la deriva, furiosas tormentas de arena, rasgos que cambian en su superficie, incuso un día de veinticuatro horas. Es tentador considerarlo un mundo habitado. Marte se ha convertido en una especie de escenario mítico sobre el cual proyectamos nuestras esperanzas y nuestros temores terrenales. El Marte real es un mundo de maravillas. Sus perspectivas futuras nos intrigan más que el conocimiento de su pasado. En nuestra época hemos escudriñado las arenas de Marte, hemos afirmado ahí una presencia, hemos dado satisfacción a un siglo de sueños.» - *Carl Sagan* - [68]

1 La palabra *planeta* viene del griego πλανητησ (planetes) y quiere decir errante/vagabundo
2 El regolito es la capa de materiales no consolidados que proceden de la memorización de la roca y otros depósitos que descansan sobre la superficie de la roca inalterada.

1.1 El Ambiente planetario

Cada planeta es un mundo y la gama de entornos planetarios es amplia, desde los relativamente benignos y planos hasta los extremadamente rocosos y hostiles. La tarea principal antes de enviar cualquier misión robótica es conocer lo mejor posible su ambiente de trabajo, tener un amplio conocimiento previo de la atmósfera y la geología del planeta mejora en gran medida las posibilidades de éxito de la misión, ya que no solo se diseñaran sistemas adecuados para el trabajo a realizar, sino que también se podrá aprovechar mejor el terreno y obtener mas y mejores datos de interés científico. Dicho esto, Marte es uno de los lugares más interesantes. Sin embargo, varios de sus procesos geológicos siguen siendo un misterio.

La superficie de Marte presenta una gran variabilidad en sus propiedades físicas, tales como en el albedo[3], composición y rugosidad de la superficie. Estas características están determinadas en gran medida por la composición a granular de Marte y los procesos geológicos que han ocurrido lo largo de su historia. Las observaciones espectroscópicas, la reflexión de radar y las operaciones superficiales han proporcionado información importante del ambientes marciano. En la tabla 1 se muestra una comparación de los parámetros de Marte y nuestro planeta Tierra.

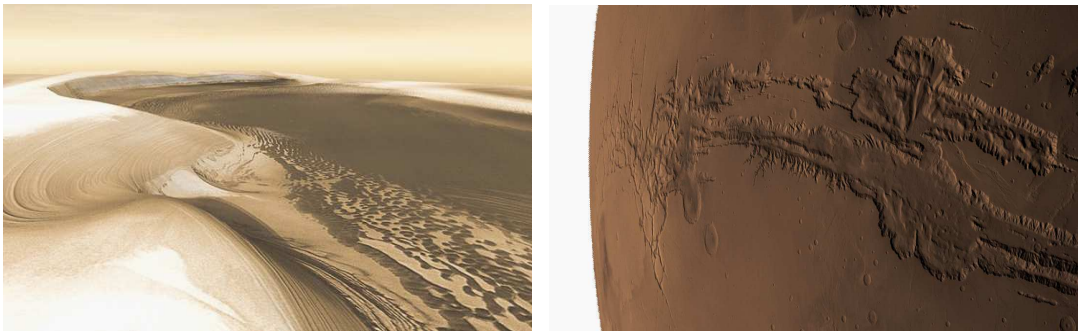
Tabla 1: Parametros fisicos de Marte. [82]

Parámetro	Marte	Tierra	Proporción Marte / Tierra
Masa [10^{24}kg]	0.64171	5.9724	0.107
Volumen [10^{10}km^3]	16.318	108.321	0.151
Densidad promedio [kg/m^3]	3933	5514	0.713
Radio del nucleo [km]	1700	3485	0.488
Radio ecuatorial [km]	3396.2	6378.1	0.532
Gravedad superficial [m/s^2]	3.71	9.80	0.379
Radiancia solar [W/m^2]	586.2	1361.0	0.431
Distancia al sol [km]	227,943,824	149,598,262	1.523
Albedo geométrico	0.170	0.434	0.392
Temperatura de cuerpo negro [K]	209.8	254.0	0.826
Pression atm. superficial [Pa]	636	101325	0.00628
Velocidad de escape [km/s]	5.03	11.19	0.450
Duración del día [hrs]	24.6597	24.0000	1.027
Número de satélites naturales	2	1	-
Anillos	No	No	-

3 El albedo es el porcentaje de radiación que cualquier superficie refleja respecto a la radiación que incide sobre ella.

Figura 1: El dios Marte y la diosa Venus (Alexandre Guillemot-1827)[58]

Aunque Marte es mucho más pequeño que la Tierra, el tamaño de su superficie terrestre no difiere mucho. Marte tiene algunos de los terrenos más variados e interesantes de todos los planetas rocosos, algunos de ellos son bastante espectaculares, por ejemplo posee el volcán más grade del sistema solar: el *Monte Olimpo* con 25 km de altura[4] y el cañón *Valles Marineris* que con sus 11 km de profundidad y 200 km de anchura es probablemente el cañón más profundo y ancho conocido del sistema solar. Los polos están cubiertos por capas de hielo que disminuyen en el verano y crecen en el invierno. La capa de hielo del norte está hecha principalmente de agua congelada[5], mientras que la capa del sur tiene una capa superior de dióxido de carbono congelado y una capa subyacente de hielo de agua. Alrededor de un 25 a 30 % de la atmósfera se condensa durante el invierno polar formando gruesos bloques de hielo de CO_2 que se subliman nuevamente cuando se exponen a la luz solar, esto crea enormes tormentas de viento desde los polos con velocidades de hasta 400 km/h[6].



(a) Cañon Chasma Boreale en el casquete polar norte de Marte

(b) Valles Marineris, el cañon más grande del sistema solar

Figura 2: Los grandes cañones de Marte[58]

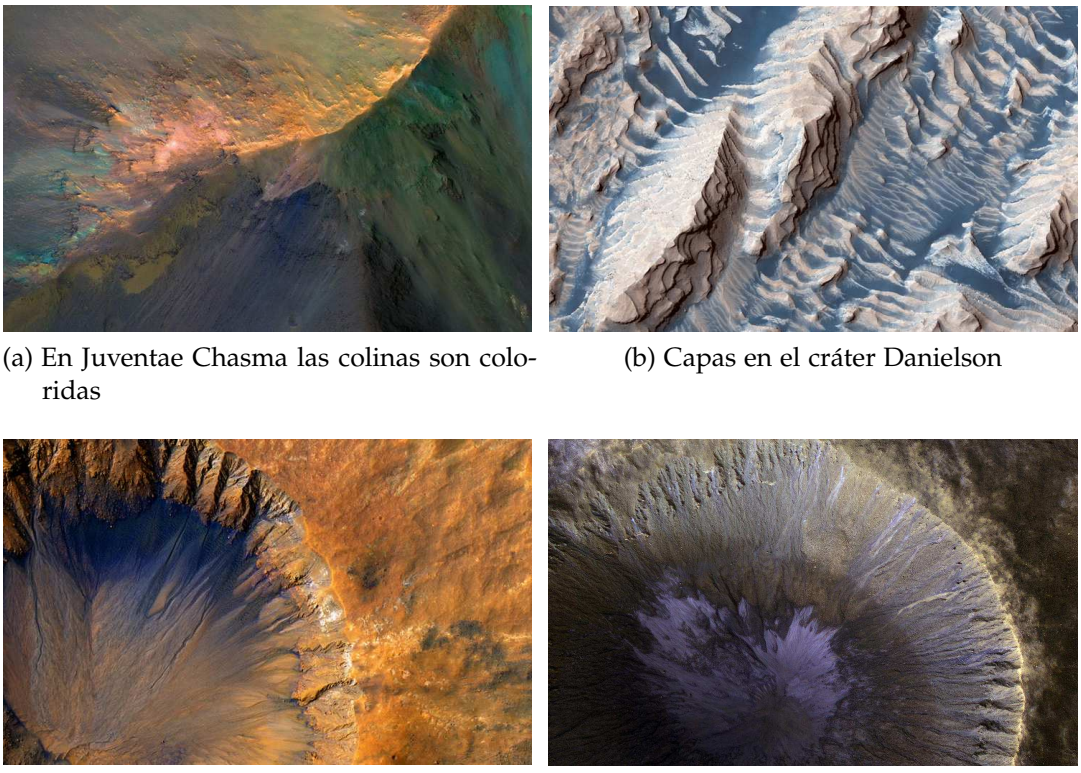
La atmósfera marciana es muy delgada, la presión en la superficie es inferior a una centésima parte de la presión atmosférica en la Tierra (es aproximadamente igual a la presión atmosférica en la Tierra a 35 km de altitud) y tiene mucha variabilidad con la altitud y la latitud. La atmósfera marciana está compuesta de 95.32 % de dióxido de carbono, 2.7 % de nitrógeno, 1.6 % de argón, 0.13 % de oxígeno, 0.08 % de monóxido de carbono, 210 ppm de agua, 100 ppm de óxido de nitrógeno, 2,5 ppm de neón, contiene 0,85 ppm de agua pesada (en forma de óxido de deuterio protio D_2O), 0.3 ppm de Criptón y trazas de metano (concentrado en unos pocos

4 El monte Everest, la montaña más alta del planeta Tierra, mide 8.848 km.

5 Se ha estimado una cifra de dos millones de km^3 de agua para la capa de hielo del norte.

lugares durante el verano septentrional), dado que el metano se descompone por radiación ultravioleta, no puede estar presente sin un mecanismo que produzca ese gas en el planeta, como la actividad volcánica, impactos de cometas o la presencia de formas de vida microbiana metanogénicas[6]. El peso molecular promedio de la atmósfera marciana es de 43.34 g/mol. [82]

Marte tiene las tormentas más grandes del sistema solar, pueden durar meses, los vientos transportan grandes cantidades de polvo, que es incluso más fino que el de la luna. El peligro para la maquinaria y los seres humanos debido al polvo rico en óxido de hierro debe tenerse en cuenta al planificar las misiones al *Planeta Rojo*. Las planicies de Marte son frecuentemente atravesadas por *dust devils* (demonios de polvo), los paneles solares de los rovers Spirit y Opportunity fueron limpiados más de una vez gracias a ellos, lo que contribuyo a mantener estos dispositivos en funcionamiento por mucho tiempo, incluso extendiendo el tiempo de vida previsto para sus misiones. La velocidad media del viento en la superficie es de 2 a 9 m/s, pero puede alcanzar ráfagas de hasta 60 m/s. Las tormentas de polvo más grandes comienzan en el sur de la primavera cuando Marte se calienta, a veces rodeando todo el planeta.



(a) En Juventae Chasma las colinas son coloridas

(b) Capas en el cráter Danielson

(c) Crater fresco cerca de la región Sirenum Fossae

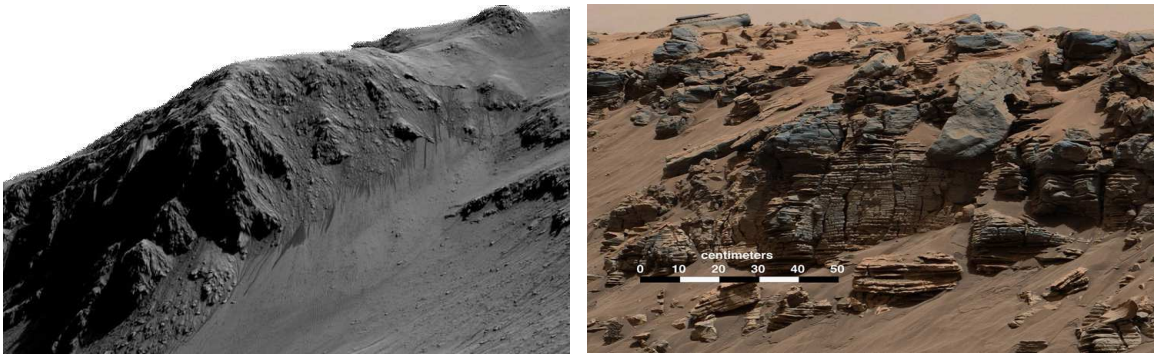
(d) Una vista invernal de un cráter con barranco

Figura 3: Imágenes aéreas de la geografía de Marte[58]

El análisis in situ de la roca marciana indica una composición de roca basáltica y andesítica enriquecida en óxidos y sulfuros. Los minerales de silicato dominantes son olivina, feldespatos de plagioclasa, piroxenos (clinopiroxeno y ortopiroxeno

6 La metanogénesis es la formación de metano por parte de los seres vivos. Es una forma de metabolismo microbiano muy importante y extendida.

en depósitos de Hesperian y Noachian, respectivamente) y cuarzo mezclado con óxidos de Ti−Fe. La composición del manto da información del pasado geológico del planeta, el manto de Marte está compuesto principalmente de olivino, basalto, andesita y dacita. Pero solo existen cantidades minúsculas de granito.



(a) Lineas de flujos recurrentes en el cráter Horowitz

(b) Signos sedimentarios en el lecho de un lago

Figura 4: Marcas y lineas estacionales de la superficie marciana[58]

1.1.1 *Lunas de Marte*

Marte posee dos lunas pequeñas que fueron descubiertas durante la oposición perihelica[7] de 1877 por Asaph Hall en el observatorio Naval de EUA. Hall se encontraba frustrando en sus intentos por descubrir cualquier satélite perteneciente a Marte y estaba listo para darse por vencido, su esposa Chloe Angeline Stickney Hall, lo convenció de seguir buscando un tiempo más, lo que llevó al descubrimiento de la luna más externa Deimos el 12 de agosto de 1877, seguida de la luna interna Phobos seis noches después. En honor a la persuasión y apoyo de su esposa, el cráter más grande de Phobos fue nombrado Stickney. [6]

Hall nombró a las dos lunas en honor a los hijos del dios griego Ares: Fobos, la personificación del miedo y Deimos, que representa el Terror.

Fobos es la más grande y más interna de las dos lunas. Tiene una masa de 1.06×10^{16} [kg] y una forma muy irregular, su densidad es de 1900 $[kg/m^3]$, lo que sugiere que Phobos es muy poroso, tal vez porque tenga una gran cantidad de hielo en su interior o es un objeto compuesto de escombros similar a muchos asteroides, los escombros que la unen están débilmente atraídas por su gravedad.

Deimos, al igual que Phobos tiene forma irregular, tiene una masa de 2.4×10^{15} [kg] y una densidad de 1750 $[kg/m^3]$, lo que sugiere que es otro cuerpo poroso. Su superficie esta llena de cráteres, los cuales producen grandes cantidades de regolito durante su formación.

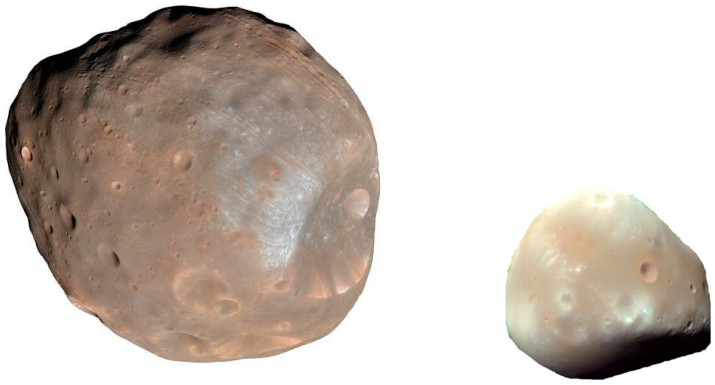
Las formas pequeñas e irregulares de Phobos y Deimos combinadas con sus superficies oscuras y similitudes espectrales con los asteroides de clase [8] C han llevado

7 La oposición es el evento en donde el Sol, la Tierra y el planeta en cuestión forman una línea recta, es el instante donde el planeta se acerca más a la Tierra y para los astrónomos, la mejor fecha para observarlo. Cuando la oposición coincide con el Perihelio, se denomina oposición perihélica.

8 Los asteroides pueden ser clasificados por sus propiedades espectrales, siendo las principales clases: S, C, M, D, V

a muchos científicos a proponer que las dos lunas son en realidad asteroides que fueron capturados por Marte. Sin embargo, las consideraciones dinámicas para un satélite capturado no pueden reproducir fácilmente las órbitas observadas de las dos lunas, lo que sugiere que las dos lunas pueden haberse formado en las proximidades de Marte.

Dentro de 20 a 40 millones de años Fobos colacionará con Marte y se formará un anillo con sus restos al rededor del planeta.



(a) Fobos (b) Deimos

Figura 5: Las lunas de Marte[58]

1.2 ¿Por qué explorar Marte?

Incluso sabiendo el misterio y curiosidad que los humanos han proyectado hacia Marte en tiempos pasados, aún surge la pregunta £por qué el interés de explorar Marte en la actualidad?, es decir, £por qué ocupa el primer lugar en la lista de los sitios a explorar?. Pues bueno, aquí hay algunas de las razones:

Después de la Tierra, Marte es el planeta con el clima más hospitalario del sistema solar. Tan hospitalario que existen grandes probabilidades de que alguna vez pudo haber albergado vida bacteriana primitiva. Sus conocidos canales de desagüe y otras características geografías arrojan evidencia de que hace millones de años fluyó agua liquida sobre su superficie, y en la actualidad existe la posibilidad de que exista agua muy por debajo de la superficie. Pero, ¿Qué cambió el clima en Marte?, ¿las condiciones necesarias para originar o albergar vida se encuentran en Marte actualmente?. Estas son algunas preguntas que impulsan a querer explorar Marte, y se suman a las grandes preguntas que surgieron no mucho después de comenzar a explorar por primera ves el universo[63]:

- ¿La vida ha ocurrido en otros planetas de nuestro sistema solar?
- ¿Cuáles son las condiciones mínimas necesarias para la formación de vida?

La búsqueda de evidencia de agua en la superficie de Marte es el principal objetivo científico, sobretodo por sus implicaciones para la vida marciana. Durante el período Hesperiano húmedo, pudo existir las condiciones adecuadas para el origen y la evolución de la vida microbiana. La vida arqueana[9] surgió rápidamente en

9 Las arqueas son un grupo de microorganismos unicelulares que, al igual que las bacterias, tienen morfología procariota.

la Tierra a principios de su historia, lo que sugiere que las arqueas pueden haber surgido en condiciones similares en Marte. Suponiendo una evolución similar a la de la Tierra con ingredientes y condiciones iniciales similares, los primeros seres en la superficie de Marte pueden haberse parecido a las cianobacterias encontradas en los registros geológicos de hace 3.5 mil millones de años en la Tierra y que actualmente dominan en los desiertos antárticos. También, se tiene la esperanza que existan lagos de agua líquida en Marte, que persistan bajo una capa protectora de hielo durante períodos significativos[17]. Esto último no esta para nada lejos de la realidad, ya que en 2018 el radar *Marsis* a bordo del orbitador *Mars Express* coordinado por la Agencia Espacial Europea (ESA), confirmó la presencia de agua líquida en lo que ellos creen es un lago de 20 kilómetros de ancho que se encuentra debajo de la capa de hielo del polo sur. Este importante hallazgo es el primer signo de existencia de agua líquida en el planeta rojo[**bbc-lake**].

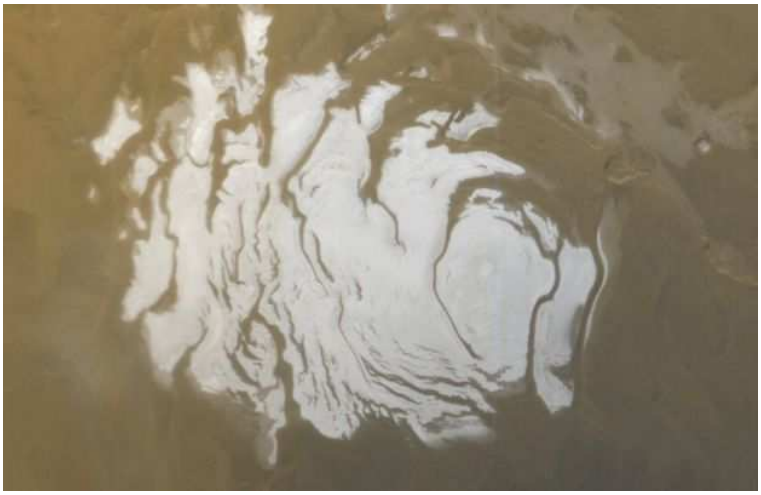


Figura 6: El lago encontrado se encuentra debajo del hielo cerca del polo sur de Marte.[58]

El interés en investigar el pasado y el presente de la vida en Marte establece que sea crucial evitar cualquier tipo de contaminación desde la tierra. La NASA y sus programa destinados a la de exploración de Marte ponen como prioridad cuatro temas de interés[63]:

- Realizar la búsqueda de evidencia de vida pasada en el planeta
- Explorar hábitats hidrotermales (los cuales mejoran la posibilidad de encontrar vida pasada y presente)
- Realizar la búsqueda de vida presente en en planeta
- Aprender sobre la evolución geológica de Marte

1.2.1 *Espeleología espacial*

Desde que realizaron las primeras fotografías de superficies planetarias en la década de 1960, es bien sabido por los geólogos planetarios que Marte y la Luna de la Tierra poseen tubos de lava, ríos sinuosos y huecos provocados por la lava que fluyó en su superficie durante periodos de vulcanismo. Sin embargo, no fue hasta el

2007, cuando el *Mars Reconnaissance Orbiter* de la NASA encontró la primera apertura a una de estas cuevas; el techo de uno de estos tubos se colapsó parcialmente creando una especie de tragaluz, que desde arriba se parece a un pozo gigante. Para el año 2009, el orbitador lunar japonés SELENE ya había encontrado estos tragaluces en la luna, así como otros pozos y orificios de diversos orígenes geológicos. Desde entonces, se han localizado cientos de cuevas más, creando un nuevo campo potencial para la ciencia: la **espeleología espacial**.

£Por qué explorar cuevas extraterrestres?, en parte por las mismas razones que los científicos se aventuran en cuevas en la Tierra, las cuevas están protegidos del clima y en otros mundos del bombardeo meteórico, por lo que sus formaciones geológicas sirven como registros inalterados del pasado del planeta. También, está la cuestión de la vida, es probable que exista agua líquida bajo la superficie de Marte, en las profundidades donde el hielo se derrite por el calor del interior del planeta.

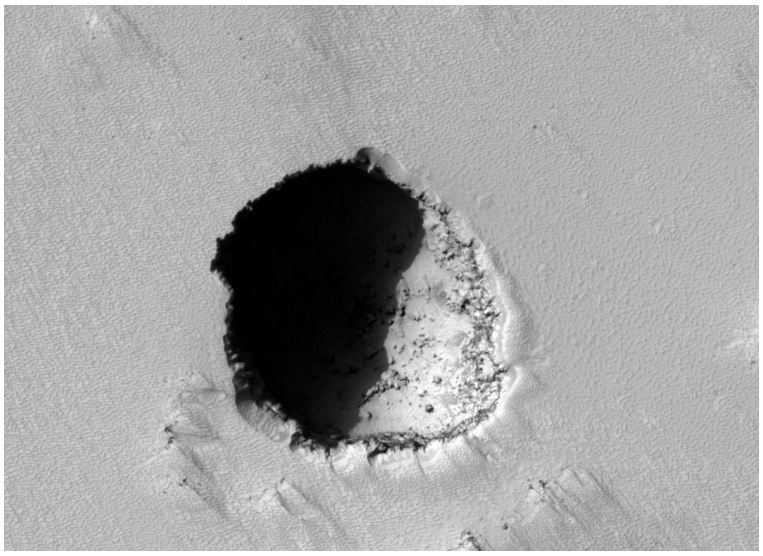


Figura 7: Pozo de 180 metros en Marte.[58]

En las cuevas, la temperatura se mantiene relativamente estable, por lo que también ofrecen un refugio contra las variaciones térmicas del día y la noche de casi 93 [deg C] de la superficie. Más importante aún una cueva ofrece protección contra la radiación, dado que el campo magnético protector de Marte se *apagó* hace eones, el constante bombardeo de rayos cósmicos en la superficie probablemente habría destruido cualquier ser expuesta. Si la vida se movió bajo tierra para escapar, las cuevas serían un buen lugar para buscar evidencia fósil, incluso es posible que en algunas cuevas marcianas la vida todavía exista. En la fig. 7 se muestra un pozo de 180 metros de ancho en Marte, probablemente es una entrada a una cueva, fue fotografiado por la cámara HiRISE de el *Mars Reconnaissance Orbiter* de la NASA[8].

Un robot que pueda escalar y navegar dentro de cavernas podría aportar excelentes capacidades a la astrobiología. Actualmente existen diferentes investigaciones relacionadas a la exploración de cuevas en otros planetas. Por ejemplo, Penny Boston, directora del Instituto de Astrobiología de la NASA, está estudiando los patrones que los microbios de la Tierra imprimen en las paredes de las cuevas, estos son

depósitos de calcio laberínticos llamados biovermiculaciones. Ella cree que estos patrones únicos nunca ocurrirían por accidente, incluso en otro planeta, si tales patrones existen en una cueva marciana, entonces es muy probable que los microbios los hayan creado y lo más convincente de estos patrones es que son una firma biológica macroscópica de la vida microscópica, por lo que pueden ser identificados y reconocidos por ejemplo, mediante métodos de visión artificial. [8]

Figura 8: Patrones creados por microbios en el interior de una cueva.[8]

Otros investigadores del JPL están probando una idea para un nuevo método de comunicación que podría permitir que un robot explorador de cuevas se comunique con su módulo de aterrizaje a través de las paredes de roca de muchos metros de espesor. Otros, prueban una técnica que podría mapear cuevas desde la órbita. Así mismo, existen proyectos multimillonarios tales como BRAILLE el cual es un proyecto financiado por el programa de Ciencia y Tecnología Planetaria de la NASA, sus misión es caracterizar los signos de vida (biológicos o químicos) dentro de las cuevas y tubos de lava.

En Marte, el enorme volcán *Monte Olimpo* ha dejado una gran cantidad de tubos de lava que rodean su base[10]. También la propia historia geológica de la Luna ha dejado una red de tubos de lava debajo de su superficie, en fin, se puede decir casi con certeza que cualquier mundo rocoso en nuestro sistema solar o más allá, ha tenido actividad volcánica en su historia. Entender la vida en los tubos de lava extraterrestres y el registro mineral que la vida misma imprime a su paso puede ser la clave para encontrar vida fuera de la Tierra, los rastros más pequeños de biología son esenciales, incluso si no hay vida actualmente, las cuevas podrían preservar evidencia de la vida antigua, solo a la espera de que los encontremos.

10 Pero se cree que el planeta Venus tiene tubos de lava abundantes y significativamente más grandes.

1.3 Misiones espaciales a Marte

Marte ha sido un destino importante para las naves espaciales desde los primeros días de la exploración espacial. Aunque hay un considerable interés en explorar Marte, este no ha sido el lugar más fácil para investigar, aproximadamente dos tercios de todas las misiones hasta la fecha han sido fracasos parciales o completos. En la tabla 2 se listan todas las misiones a Marte que han sido lanzadas hasta la actualidad.

A partir de las misiones exitosas en el planeta rojo, se han obtenido muchas fotografías de Marte, algunas por medio de satélites y otras del terreno a nivel superficial utilizando landers y rovers, con ellas se han creado bases de imágenes extensas con las cuales se ha podido conocer mejor la geología del planeta. Por lo tanto, estas bases contienen información útil con la cual se pueden planear nuevas misiones de exploración y diseñar nuevos sistemas e instrumentos adecuados a las condiciones ambientales y geográficas. Las imágenes también tienen otra utilidad especifica, y es usarlas para entrenar sistemas de navegación autónoma y reconocimiento de terreno mediante *inteligencia artificial*, con esto los rovers podrán realizar tareas de evasión de obstáculos, localización y mapeo simultaneo (SLAM, por sus siglas en inglés), obtención de muestras, y mucho más de manera autónoma.

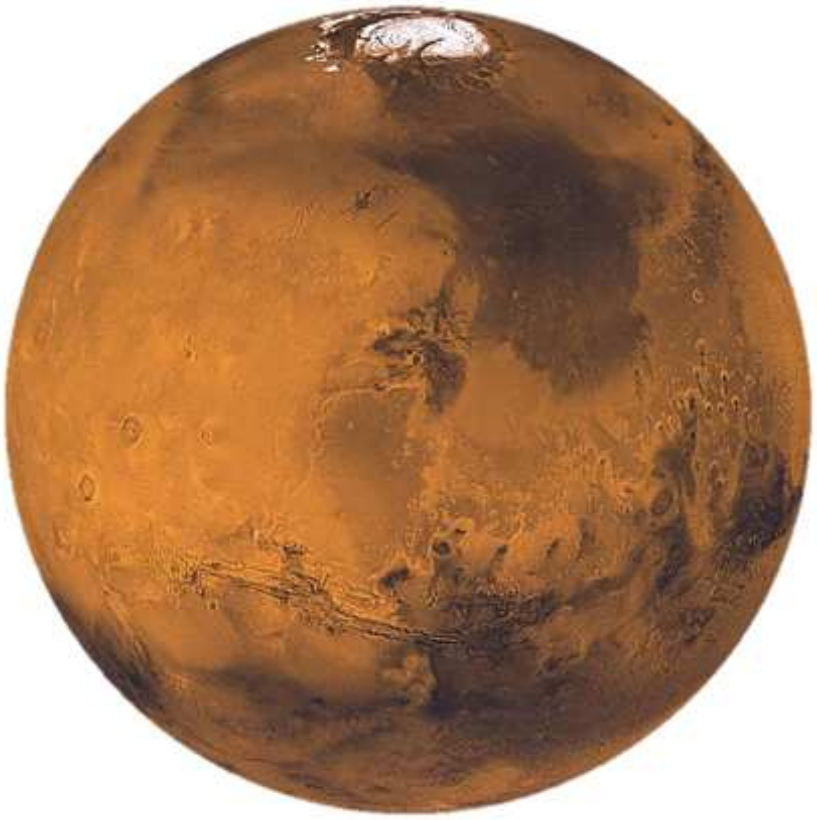


Figura 9: Globo Marciano, imagen compuesta por el satélite Viking 1[59].

Tabla 2: Misiones espaciales a Marte [6]

Misión	País	Fecha	Tipo de misión	Anotaciones
Sin nombre	URSS	10/10/60	Sobrevuelo	No alcanzó la órbita terrestre
Sin nombre	URSS	14/10/60	Sobrevuelo	No alcanzó la órbita terrestre
Sin nombre	URSS	24/10/62	Sobrevuelo	Solo alcanzó la órbita terrestre
Mars 1	URSS	1/11/62	Sobrevuelo	El radio falló a 106×10^6 km
Sin nombre	URSS	4/11/62	Sobrevuelo	Solo alcanzó la órbita terrestre
Mariner 3	EUA	5/11/64	Sobrevuelo	Fallo de la cubierta protectora
Mariner 4	EUA	28/11/64	Sobrevuelo	Voló con éxito el 14/7/65
Zond 2	URSS	30/10/64	Sobrevuelo	Pasó Marte pero fallo la radio
Mariner 6	EUA	24/2/69	Sobrevuelo	Voló con éxito el 31/7/69
Mariner 7	EUA	27/3/69	Sobrevuelo	Voló con éxito el 8/5/69
Mariner 8	EUA	8/5/71	Orbitar	Falló durante el lanzamiento
Kosmos 419	URSS	10/5/71	Lander	Solo alcanzó la órbita terrestre
Mars 2	URSS	10/5/71	Orbitar/Lander	El módulo de aterrizaje falló
Mars 3	URSS	28/5/71	Orbitar/Aterrizaje	llegó el 12/3/71; algunos datos
Mariner 9	EUA	30/5/71	Orbitar	órbita del 11/13/71 al 10/27/72
Mars 4	URSS	21/7/73	Orbitar	Fracasó, pasó Marte el 2/10/74
Mars 5	URSS	25/7/73	Orbitar	Llegó el 2/12/74; duró días
Mars 6	URSS	5/8/73	Orbitar/Lander	Llegó el 3/12/74; pocos datos
Mars 7	USSR	9/8/73	Orbitar/Lander	Llegó el 3/9/74; pocos datos
Viking 1	EUA	20/8/75	Orbitar/Lander	Operado por 2245 soles
Viking 2	EUA	9/9/75	Orbitar/Lander	Operado por 1281 soles
Phobos 1	URSS	7/7/88	Orbitar/Lander	Perdido en ruta a Marte
Phobos 2	URSS	12/7/88	Orbitar/Lander	Perdido en Marzo de 1989
Mars Observer	EUA	25/9/92	Orbitar	Perdido justo antes de llegar a Marte
Mars Global Surveyor	EUA	7/11/96	Orbitar	Operdo del 9/12/97 al 11/2/06
Mars 96	Rusia	16/11/96	Orbitar/Lander	Fallo del lanzador
Mars Pathfinder	EUA	4/12/96	Lander/rover	Operado del 7/4/97 al 9/27/97
Nozomi	Japón	4/7/98	Orbitar	En órbita heliocéntrica
Mars Climate Orbiter	EUA	11/12/98	Orbitar	Perdido a la llegada el 9/23/99
Mars Polar Lander/ Deep Space 2	EUA	3/1/99	Lander/ penetrar	Perdido a la llegada el 12/3/99
Mars Odyssey	EUA	7/4/01	Orbitar	Llegó el 10/24/04; sigue operando
Mars Express	ESA	2/6/03	Orbitar/Lander	Legó el 12/25/03; orbitador operando, lander perdido al aterrizar
Spirit	EUA	10/6/03	Rover	llegó el 1/3/04; aún operando
Opportunity	EUA	7/7/03	Rover	Llegó el 1/25/04; aún operando
Rosseta	ESA	2/3/2004	Asist. gravitatoria	Sobrevuelo en febrero de 2007 en ruta a 67P/ChuryumovGerasimenko
Mars Reconnaisance Orbiter	EUA	12/8/05	Orbitar	Llegó el 3/10/06; aún operando
Phoenix	EUA	4/8/2007	Lader	Operó de mayo a noviembre de 2008
DAWN	EUA	27/9/2007	Asist. gravitatoria	Sobrevuelo en febrero de 2009 en ruta a 4 Vesta y Ceres
Fobos-Grunt	Rusia	8/11/2011	Orbitar	Falló, nuca abandonó la órbita LEO
Yinghuo-1	China	8/11/2011	Orbitar	Debió desplegarlo Fobos-grunt
Curiosity (MSL)	EUA	26/11/2011	Rover	En operación desde el 6/8/2012
Mars Orbiter Mission	India	5/11/2013	Orbitar	En operación
MAVEN	EUA	18/11/2013	Orbitar	En operación
ExoMars Trace Gas Orbiter	ESA/Rusia	14/6/2016	Orbitar	En operación
Schiaparelli EDM lander	ESA	14/3/2016	Lander	Se estrelló
InSight	EUA	5/5/2018	Lander	En operación
MarCO	EUA	5/5/2018	CubeSats	Último contacto en febrero de 2019
Perseverance (Mars 2020)	EUA	30/7/2020	Rover/helicoptero	Viajando rumbo a Marte

1.4 Base de imágenes PDS

El sistema de datos planetarios (PDS, por sus siglas en inglés) es un archivo de datos digitales que pertenece a la NASA, fueron obtenidos de las misiones planetarias *Mars Exploration Rover* (MER) y *Mars Science Laboratory* (MSL) de la NASA. Los Archivos están disponibles en línea y son abiertos al publico. El PDS tiene un nodo de imágenes llamado Atlas de Imágenes Planetarias (PIA, por sus siglas en inglés) el cual es una aplicación que provee de imágenes planetarias a la comunidad científica y al publico en general. El Atlas fue diseñado con la capacidad de buscar y filtrar más de 8 millones de archivos de imágenes planetarias.[31]

En el PIA se pueden encontrar más de 20 millones de imágenes de Marte, muchas de ellas tomadas con las cámaras de los rovers durante sus misiones. Las imágenes pueden filtrarse por misión. instrumentos, nave, objetivo, etc. En la fig. 10 se muestra una captura de pantalla del buscador.

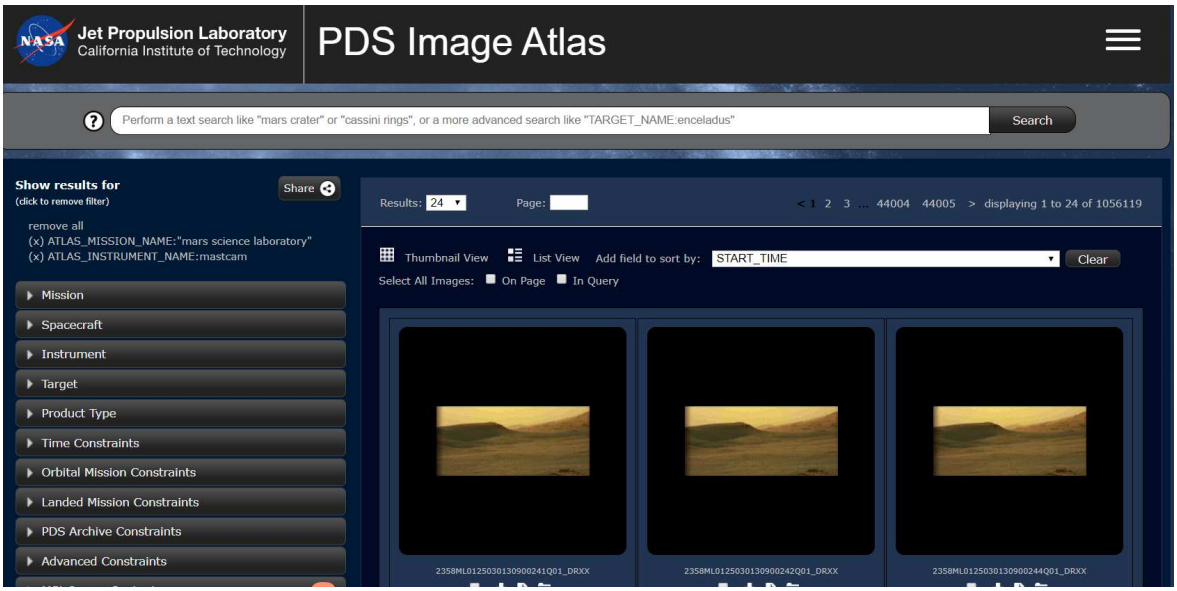


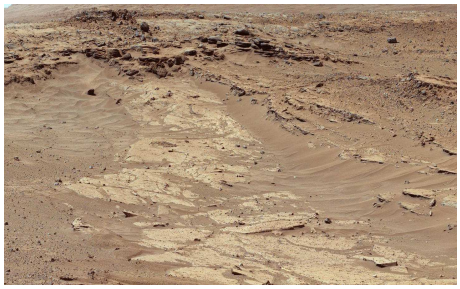
Figura 10: Atlas de imágenes planetarias (A la izquierda menú de filtrado)[31]

Otra fuente para obtener imágenes sin procesar de las misiones rover de la NASA es la base de imágenes sin procesar del *Mars exploration Program*[56] donde se pueden encontrar miles de imágenes de las cámaras PanCams, Hazcams, NavCams y demás cámaras científicas instaladas en el rover Curiosity, que hasta el día de hoy, después de mas de 2500 soles, sigue enviando imágenes interesantes del terreno marciano. En la figura 11 se muestran imágenes del terreno marciano de gran interés geológico, obtenidas gracias a las misiones con robots en el planeta rojo.

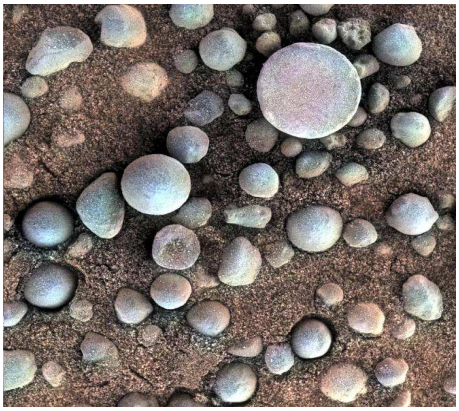
(a) Erosión por retirada de escarpa

(b) Monte Sharp

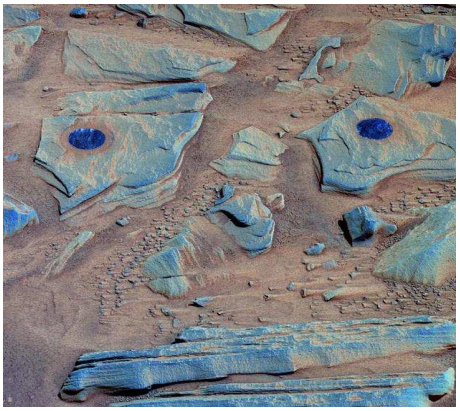
(c) Cráter Victoria



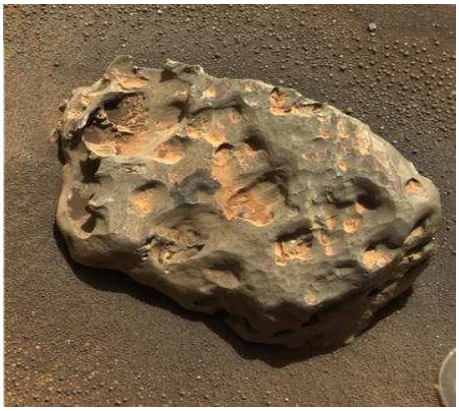
(d) Erosión diferencial



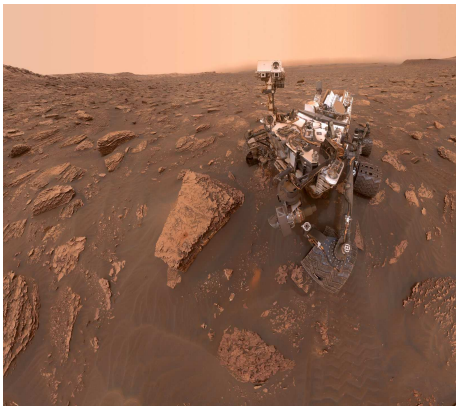
(e) Arándanos marcianos



(f) Afloramiento rocoso



(g) Meteorito de hierro



(h) Curiosity en Duluth

Figura 11: Imágenes de la variedad geológica en marte[58]

2

ROBOTS EXPLORADORES

Este capítulo presenta una visión general de la robótica espacial, así como los problemas abiertos de la robótica móvil con aplicaciones espaciales. Por otro lado, se trata el tema de los robots exploradores (rovers), se describe sus principales sistemas. También, el capítulo introduce el uso de las redes neuronales para el reconocimiento de terreno y su utilidad en la exploración espacial. Para finalizar, se describen las misiones exitosas de los rover en Marte: *Mars Exploration Rover* y *Mars Science Laboratory* y la misión más reciente *Mars 2020*.

La palabra **robot** deriva del término checo ROBOTNIK; fue tomado de la obra de teatro de 1920 llamada RUR (Rossum's Universal Robots), obra del dramaturgo y novelista checo `Karel apek`, literalmente significa *trabajador forzado*. A través de los años las múltiples definiciones de lo que es un robot se han quedado cortas debido a la gran variedad de aplicaciones de estas máquinas, así como su amplia diversidad de configuraciones. Según el diccionario Merriam-Webster, un robot es:

« *Un dispositivo automático que realiza funciones normalmente atribuidas a humanos o animales; una maquina construida para parecerse a un ser humano o animal en apariencia y comportamiento.*» [14]

La definición de Webster ha sido una de las más utilizadas porque cubre ampliamente las tareas y las funciones robóticas, más allá de solo mover cosas. Los *robots móviles* son un subconjunto de los robots en general y poseen la principal característica de que pueden moverse en un entorno determinado (a diferencia de los robots industriales de base fija), un robot móvil no ve limitada su movilidad debido a su tamaño físico, como resultado, los robots móviles pueden operar en un gran espacio de trabajo y explorar entornos desconocidos. Para propósitos de este trabajo, se propone la siguiente definición general de lo que es un robot, la cual abarca en lo posible, la gran variedad de tecnologías, plataformas y aplicaciones conocidas actualmente de estos dispositivos:

Definición 2.0.1. Robot: *Maquina automática u operada que asemeja el comportamiento y apariencia de los seres vivos, o una parte de ellos, con el fin de realizar acciones complejas o alguna tarea específica.*

En la última década, ha existido un gran crecimiento en la investigación de la robótica móvil, esto ha originado diversos campos de estudio que cuentan con su propia terminología bien definida. De manera general, los robots móviles pueden ser clasificados en tres categorías según sus entornos operativos: 1) robots terrestres, 2) robots submarinos y 3) robots aéreos. No obstante, aunque operan en entornos diferentes, todo sistema robótico posee un conjunto de partes funcionales básicas que son análogas a las partes de cualquier ser vivo[11].

Un modelo funcional que ilustra lo anterior se muestra en la Fig. 12, aquí se puede observar que el sistema se compone por la unión de diferentes bloques funcionales, donde la dirección de flujo de energía y la trayectoria de retroalimentación de información se ilustra mediante flechas. En el modelo se observa como el bloque intelectual controla el bloque de actuación, y a su vez este controla el sistema de movilidad, el cual está conformado de estructuras y piezas motivacionales[1] impulsadas por actuadores. La estructura se refiere al armazón del cuerpo (esqueleto en el caso de un ser humano) y al armazón mecánico de un robot. Las piezas motivacionales son elementos que promueven o causan el movimiento (músculosx en los sistemas orgánicos), la movilidad también engloba las piezas mecánicas accesorias para los robots, estas amplían o refinan la movilidad del robot para ejecutar tareas específicas, tales como las ruedas, orugas y hélices que aumentan el alcance del movimiento del robot mucho más allá de su estructura, las pinzas y los efectores finales mejoran la destreza del espacio de trabajo y su manipulación.

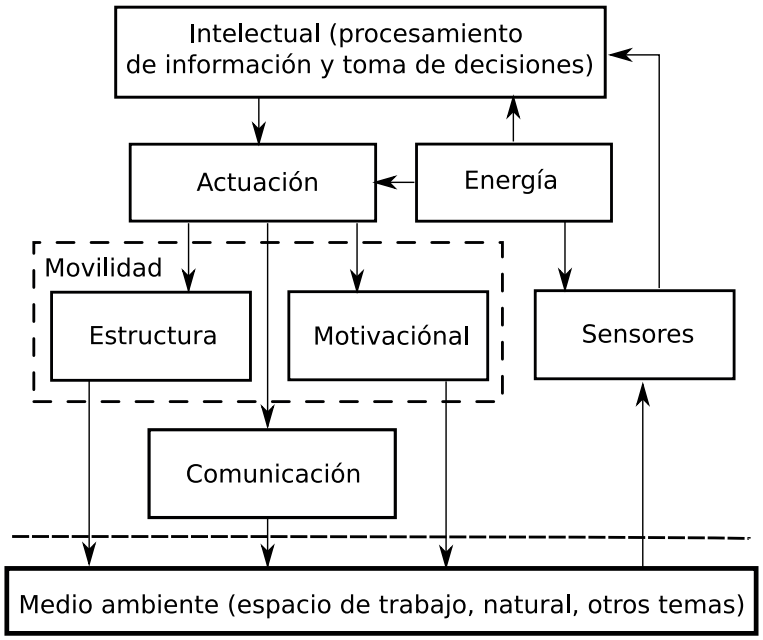


Figura 12: Modelo funcional generalizado para robots[11].

El bloque sensorial comprende todo aquello por lo cual el robot puede obtener información del mundo exterior; puede ser tan sencillo como un arreglo de sensores o tan complejo como todo un sistema de visión que incluya varias cámaras y algoritmos de reconocimiento, esto depende del entorno de trabajo y de la información previa que posea el robot. Los entornos operativos pueden clasificarse en las siguientes tres categorías según la información a priori del robot: *Entorno predefinido y estructurado*. El robot tiene todo el conocimiento sobre el medio ambiente y los objetos a tratar. *Entorno semiestructurado*. El robot tiene algunos conocimientos previos (por ejemplo, mapas GPS) sobre el entorno, pero este puede cambiar espacial y temporalmente. *Entorno no estructurado*. El robot no tiene conocimiento a priori sobre su ambiente de trabajo, él tiene que confiar en su potente sistema sensorial.

1 Entiéndase el término motivacional proveniente del latín motivus (movimiento), el sufijo -ción (Acción y efecto) y el sufijo -al (relativo a), es decir, realativo a algo que provoca la acción de movimiento.

Tanto los robots como los seres humanos poseen el mismo conjunto de bloques funcionales, por lo tanto entre ellos existe una correspondencia para cada bloque. En la Tabla 3 se muestra una comparación entre los robots y el ser humano en términos de realización de sus funciones. Es evidente que para que un robot pueda realizar tareas similares a las de un humano, debe poseer los siete bloques funcionales básicos.

Tabla 3: Robots y seres humanos: bloques funcionales. [11]

Bloques funcionales	Humanos	Robots
Intelectual	Sistema nervioso	Microprocesador (Hardware y software)
Estructural	Esqueleto	Bastidor Mecánico (fuselaje, chasis, carcasa)
Motivacional	Extremidades	Ruedas, patas, orugas, hélices, pinzas, etc.
Actuación	Músculos	Actuadores hidráulicos, eléctricos, neumáticos. piezoeléctricos, electrostáticos; músculos artificiales
Percepción sensorial	Ojos, orejas, piel, boca, etc.	Cámaras, sensores ópticos, sonar, sonido, luz infrarroja, campos magnéticos, radiación, etc.
Comunicación	Palabras, gestos, ademanes	Datos, imagen / video, sonido
Energía	Alimentos, glucógeno	Fuente de poder, baterías

Un robot móvil es un conjunto complejo de bloques fundamentales, cada uno de los cuales debe elegirse cuidadosamente en función de las especificaciones de terreno, duración de la misión, objetivo y condiciones atmosféricas. Un diseño óptimo tiene como objetivo que el robot cumpla una misión específica de la manera más rentable y eficiente. Los robots son ampliamente utilizados en las misiones espaciales, los tipos de misiones que requieren el uso de robots y manipuladores en el espacio se subdividen en los siguientes tipos[17]:

1. Misiones robóticas de exploración
2. Misiones robóticas comerciales y de explotación de recursos
3. Misiones robóticas para preparar el camino para futuras misiones humanas
4. Misiones de exploración humana con dispositivos robóticos como apoyo y exploración
5. Misiones de exploración humana con vehículos para mejorar la movilidad
6. Misiones humanas de exploración/explotación que requieren dispositivos de construcción y excavación.

Los robots en el entorno espacial proyectan la influencia humana donde los humanos aun no pueden ir, en el caso de misiones no tripuladas un factor importante a tomar en cuenta es la lejanía de la Tierra a la que se debe operar los dispositivos. Se debe tomar en cuenta la demora en la comunicación, no es solamente por la distancia que toma (2 a 3 segundos para la Luna, y más de media hora para Marte), sino también debido a la disponibilidad de los transmisores de radio en órbita alrededor del cuerpo celeste y de la Tierra.
Los robots espaciales se pueden utilizar en tres tipos distintos de entornos:

- Órbita terrestre baja (LEO)
- Espacio profundo
- Superficies planetarias

ORBITA LEO: Los robot utilizados en orbitas LEO incluyen las naves espaciales autónomas, así como los brazos robóticos (tele-manipuladores) auxiliares en actividades extra-vehiculares y trajes espaciales robóticos. Muchos satélites no tripulados que operan en órbita baja funcionan sin o con muy poca intervención humana. Sin embargo, la mayoría de los satélites científicos y comerciales, aunque operan de manera autónoma se les considera más máquinas automáticas que robots[45], ya que operan en líneas fijas realizando tareas repetitivas. Solo los satélites muy complejos pueden considerarse robots, por ejemplo, el telescopio espacial Hubble.

ESPACIO PROFUNDO: Las sondas de espacio profundo generalmente se consideran robots ya que deben realizando una amplia variedad de tareas de manera autónoma. Mientras más lejos de la tierra deba operar un robot, mayor deberá ser su nivel de autonomía. Un ejemplo son las sondas espaciales Voyager 1 y 2 que contienen generadores eléctricos nucleares que han permitido que sus instrumentos cientificos se encuentren funcionando desde hace 42 años, actualmente se encuentran a una distancia de la tierra de 149 y 123 [UA] respectivamente (aproximadamente 22 Mil millones de kilometros), lo que los convierte en los instrumentos artificiales más lejanos jamás enviados por el ser humano.

SUPERFICIES PLANETARIAS: Los robots utilizados hasta ahora en superficies planetarias se pueden dividir en dos tipos: los módulos de aterrizaje (lander) y los robots exploradores (rover). Los módulos de aterrizaje son básicamente una nave que desciende y se posa sobre la superficie de un cuerpo celeste, realiza un aterrizaje suave (a diferencia de una sonda de impacto), posteriormente el lander sigue funcionando y realiza tareas especificas. En la actualidad, los landers son capaces de realizar muchas tareas diferentes, como desplegar rovers, tomar muestras de terreno, realizar experimentos científicos y enviar imágenes a la Tierra. Los módulos de aterrizaje futuros podrán realizar muchas otras tareas como: desplegar aviones o globos de exploración, recuperar vehículos para devolver muestras, o incluso explotar los recursos locales. Los rovers, por otro lado, son vehículos de exploración espacial diseñados para moverse sobre la superficie de un planeta u otro cuerpo astronómico. Algunos rover han sido diseñados para transportar tripulantes tales como el *Lunar Roving Vehicle* (LRV) de las misiones Apolo 15,16 y 17.

2.1 Problemas abiertos de la robótica

Muchos de los problemas de la robótica espacial no difieren mucho de los de la robótica móvil *convencional*, pero a menudo son llevados a niveles más altos. En un futuro la producción a gran escala hará posible realizar estudios muy profundos en el área de la robótica a un costo cada vez más razonables; la robótica espacial se beneficiará de la transferencia tecnológica de aplicaciones no espaciales con alta tecnología, y a cambio esta última encontrará en la aplicación espacial un conducto

para las soluciones más exigentes[45].

Según [46], La investigación de robótica artificialmente inteligente se puede clasificar en tres ramas principales: La primera rama intenta obtener conocimiento de los procesos inteligentes del cerebro (aprendizaje, pensamiento, razonamiento, etc.), la segunda se centra en proporcionar observabilidad y controlabilidad mediante el modelado de sensores, la fusión de datos, la teoría de control y otros temas relacionados, por ultimo, la tercer rama comprende esfuerzos para crear algoritmos manejables para resolver los problemas identificados en las dos primeras ramas.

La mayor limitación a la que se enfrentan los investigadores es la *potencia de cálculo*, cuando se compara con el poder computacional del cerebro humano, la capacidad de las computadoras actuales es muy limitada. Esta limitación impide lograr un rendimiento en tiempo real, incluso si se tuviera una comprensión y un conocimiento completo de los procesos inteligentes del cerebro para emularlos.

También, los factores que complican aún más los algoritmos de navegación robótica al aire libre incluyen el problema de reconocimiento de objetos, detectar la presencia de obstáculos (posiblemente aleatorios) y las incertidumbres generadas por las múltiples modalidades de detección utilizadas por un robot, así como las limitaciones de los dispositivos de detección en sí y una teoría deficiente.

PROBLEMAS DE CONTROL: Tanto el hardware como el software de control necesitan aún mucho desarrollo. El hardware se está desarrollando rápidamente, pero los dispositivos más recientes y potentes aún no están calificados para el entorno espacial, especialmente en lo que se refiere a la radiación; por lo tanto, los robots espaciales deben confiar en un hardware menos poderoso y más anticuado que sus homólogos de la Tierra. El software también es crítico, particularmente para aplicaciones no convencionales, como robots altamente autónomos, dispositivos cooperativos y telemanipuladores, así como para interfaces hombre-máquina que se pueden usar en entornos extremos.

PROBLEMAS EN MECANISMOS: Los componentes mecánicos y estructurales de los robots espaciales son incluso más complejos que los de los robots convencionales. el bajo peso y alta confiabilidad son requisitos criticos, así como la posibilidad de trabajar en entornos extremos. Un robot espacial debe soportar una fuerte vibración en la etapa de lanzamiento y en el reingreso/aterrizaje, el vacío del espacio, problemas de lubricación y grandes variaciones de temperatura, radiación, polvo durante largos períodos de tiempo sin mantenimiento; esto es un gran reto para todos los sistemas mecánicos que tienen una gran probabilidad de falla por su misma naturaleza de funcionamiento. Así mismo, también es necesario trabajar mucho en el área de las estructuras desplegables.

PROBLEMAS EN LOS TRANSDUCTORES: Los sensores y actuadores son componentes críticos en cualquier sistema robótico. Si bien los sensores pueden ser similares a los de los robots estándar, en el espacio el uso de diferentes tipos de actuadores está muy limitado por el medio ambiente y por la poca potencia disponible. Algunos tipos de actuadores no se pueden utilizar en absoluto, tales como

los motores de CC con escobillas que no funcionan correctamente en el vacío y las aleaciones con memoria de forma (SMA, por sus siglas en inglés) que necesitan refrigeración y tienen una eficiencia muy baja. También, los dispositivos neumáticos e hidráulicos rara vez son considerados, los primeros por su alto consumo de aire y baja eficiencia y los segundos por la necesidad de tener equipos auxiliares de gran peso. Los actuadores considerados para la mayoría de las aplicaciones espaciales se restringen a motores eléctricos (generalmente sin escobillas) y actuadores electromagnéticos o piezoeléctricos.

PROBLEMAS DE POTENCIA: La potencia es un problema muy crítico en todas las aplicaciones robóticas. Los transductores y la electrónica de control generalmente están energizados por baterías, pero estas deben mantenerse en buenas condiciones de carga; en misiones muy cortas a menudo se utilizan baterías primarias (no recargables). La solución más común es el uso de paneles solares, pero las tecnologías actuales poseen baja eficiencia y su efectividad disminuye rápidamente al alejarse del Sol. En Marte por ejemplo, su producción de energía es de la mitad que en la Tierra, y prácticamente no pueden usarse (más allá de la órbita de Júpiter).

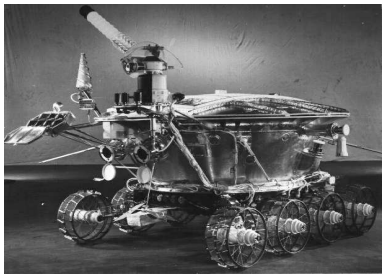
Los generadores termoeléctricos de radioisótopos (**RTG!**, por sus siglas en inglés) son una opción para suministrar poca energía durante mucho tiempo y por lo general son la mejor opción para sondas que operan en el espacio profundo. No obstante, tienen algunas limitaciones en aplicaciones planetarias. Es probable que los reactores nucleares sean la mejor opción para producir grandes cantidades de energía para el espacio profundo y las instalaciones planetarias.

PROBLEMAS EN COMUNICACIONES: La necesidad en comunicaciones depende de la aplicación y tipo maquinas consideradas, por ejemplo los tele manipuladores necesitan recibir todos los comandos en tiempo real (o casi real); cuanto más autónomo sea el robot, menor será la necesidad de recibir comunicaciones. La tecnología actual permite enviar información desde las fronteras de nuestro sistema solar aún con la baja potencia disponible a bordo de las sondas espaciales, sin embargo, esto requiere de equipos de recepción grandes en la Tierra.

2.2 Rovers

El término rover literalmente significa *vagabundo*, fue acuñado durante las misiones Apolo de la NASA para designar sus vehículos lunares tripulados; Este término ya había sido utilizado en el pasado para una clase de automóviles construidos en Gran Bretaña[17]. No obstante, su popularidad en las misiones a Marte, ha hecho que en la actualidad sea imposible no relacionarlo inmediatamente con la exploración del espacio. Los rovers son especialmente útiles para casi todo tipo de misiones planetarias u otros cuerpos con superficies sólidas, ya sean pequeños asteroides, cometas o la lunas de los gigantes gaseosos. Debido a las dificultades de la teleoperación es necesario que los rovers cuenten con un buen grado de autonomía, no tenerlo provocaría un funcionamiento lento e ineficiente. [45].

A través del tiempo se ha desarrollado una amplia variedad de sistemas móviles para los rovers, con diversas geometrías, tamaños y configuraciones (ver fig. 13). Estos sistemas comparten diferentes cualidades de rendimiento bajo condiciones operativas especificas. Sin embargo, aunque muchas de estas configuraciones se han probado en condiciones simuladas en el planeta Tierra, pocas han sido realmente utilizados en misiones planetarias reales, a diferencia de por ejemplo el sistema de suspensión Rocker-bogie de la NASA, que ha dado tan buenos resultados en campo que se ha convertido en el referente para los rover exploradores.

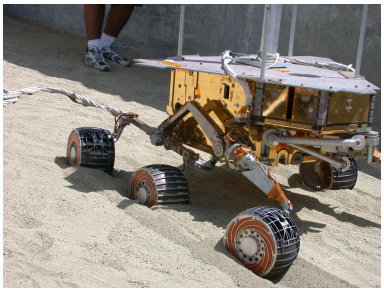


(a) Lunokhod 1 (URSS)

(b) Nanokhod (ESA)

(c) Cuatro orugas (JAXA)

(d) SCORPION (DARPA)



(e) Rocker-bogie (NASA)

Figura 13: Algunos ejemplos de sistemas móviles[72]

Los rovers con orugas, patas o dispositivos de salto se han probado en la Tierra, así como otros vehículos de exploración basados en fuerzas aerodinámicas (aviones) o fuerzas aerostáticas (globos). Los rovers pueden clasificarse sistemáticamente según su tipo de movilidad de la siguiente manera[72]:

1. Sistemas aéreos
2. Sistemas deslizantes
3. Sistemas rodantes
4. Sistemas habilitados para ruedas
5. Sistemas habilitados para piernas
6. Sistemas habilitados para seguimiento
7. Sistemas de salto
8. Sistemas híbridos

El tamaño de los rovers también es variado, desde menos de un metro hasta tamaños tan grandes como los de un camión, los más grandes están planeados para que en un futuro tengan la capacidad de configurar hábitats, reactores nucleares u otro tipo de infraestructura, tal como pistas de aterrizaje, carreteras, etc. Por lo tanto, debido a la variedad de tamaños que pueden presentar, resulta más sencillo clarificarlos en términos de masa de la siguiente manera:

- Nanorovers (masa menos de 5 kg)
- Microrovers (masa entre 5 y 30 kg)
- Minirovers (masa entre 30 y 150 kg) y
- Macrorovers (masa por encima de los 150 kg)

2.2.1 *Subsistemas de los rovers*

POTENCIA: La principal fuente de energía proviene de un arreglo de paneles solares, cuando se encuentran completamente iluminados los paneles generan aproximadamente 140 Watts durante cuatro horas/sol en Marte[2][**mission-rover**]. El rover necesita alrededor de 100 Watts para funcionar, la potencia adicional se utiliza para los experimentos científicos. El sistema de alimentación del *Mars Exploration Rover* incluye dos baterías recargables que proporcionan energía cuando el sol se oculta.

ESTRUCTURA: El cuerpo móvil recibe el nombre de Warm Electronic Box (WEB), La carrocería del rover consiste en una capa que protege la conmutadora, la electrónica y las baterías (el cerebro y el corazón del rover). La temperatura en las noche marciana puede caer hasta $-93°$C, sin embargo los componentes internos no superan el rango de $-40°$ a $+40°$ C, esto se logra mediante diferentes métodos como: pintura dorada, aerogel y calentadores eléctricos o de radio-isotopos.

MOVILIDAD: La movilidad es una capacidad vital para las misiones espaciales debido a que aumenta el retorno de información valiosa de diferentes sitios a diferencia de los Landers que son estáticos. A través de los años el desarrollo tecnológico ha dado lugar a numerosos sistemas móviles. Algunos de los sistemas son derivados de aplicaciones terrestres como los automóviles que utilizan ruedas, tanques que utilizan orugas y globos aéreos, etc.

COMPUTADORA DE ABORDO: La computadora (el cerebro del rover) yace dentro de un módulo llamado *Módulo Electrónico Rover* (REM, por sus siglas en inglés) que a su vez esta dentro del cuerpo móvil. La computadora principal utiliza una interfaz de comunicación llamada bus VME que permite el intercambio de datos con los instrumentos y sensores del móvil, este es un bus de interfaz estándar industrial para comunicarse y controlar todos los motores, instrumentos científicos y funciones de comunicación.

2 El término *sol* es utilizado por los ingenieros de la NASA para referirse a un día solar en Marte. En la Tierra un día tiene 24 horas, en cambio en Marte un sol dura 24 horas y 39 minutos.

La computadora contiene una memoria especial tolerante a la radiación protegida contra los ciclos de prendido y apagado, permite que los programas y los datos permanezcan y no se borren cuando el vehículo se apaga por la noche. Los rovers Spirit y Opportunity contenían una memoria de 128 MB DRAM con detección y corrección de errores y 3 MB de EEPROM, mientras que el rover Curiosity tenía 8 veces más (256 MB de DRAM y 2Gb de memoria Flash) el procesador funciona normalmente a una velocidad de hasta 200 MHz

INTELIGENCIA ARTIFICIAL: Los Rovers exploradores de Marte utilizan formas básicas de control para realizar ciertas tareas de forma autónoma en la superficie de Marte. Los científicos e ingenieros toman varios enfoques para controlar los robots. Los dos extremos del espectro son el *control deliberativo* y el *control reactivo*. El primero es la forma tradicional y dominante como que funcionan los robots, construyendo minuciosamente los mapas y otros tipos de modelos que utilizan para planificar secuencias de acción con precisión matemática.

El inconveniente de esto es que si algo interrumpe el progreso del robot (por ejemplo, si el mapa es incorrecto o carece de detalles), el robot debe detenerse, crear un nuevo mapa y un nuevo plan de acción. Este proceso de re-planificación puede ser costoso si se repite con el tiempo. Además, para garantizar la seguridad del robot, se deben implementar programas de respaldo para abortar el plan si el robot se encuentra con una roca o un agujero imprevisto que pueda dificultar su viaje.

Los *enfoques reactivos* por otro lado, eliminan los mapas y la planificación por completo y se centran en la observación del entorno en tiempo real. Bajar la velocidad si hay una roca por delante. Cavar si hay una marca en el suelo, etc.

El grupo de Homayoun Seraji del JPL se centra en dos de los muchos enfoques para implementar el control basado en el comportamiento: la *lógica difusa* y las *redes neuronales* artificiales. La principal diferencia entre los dos sistemas es que los robots que usan lógica difusa se desempeñan con un conocimiento establecido, cercano al modo de pensar de los seres vivos, pero que no mejora con el tiempo, mientras que los robots con redes neuronales comienzan sin conocimiento y aprenden con el tiempo.

Las redes neuronales artificiales son modelos computacionales que permiten a los robots aprender de sus experiencias, asociar percepciones con acciones y adaptarse a situaciones o entornos imprevistos. Los rovers pueden ser entrenados por humanos, en un entorno de laboratorio para reconocer y moverse en terrenos difíciles, la habilidad aprendida puede ser aplicada cuando el robot se encuentra en un sitio desconocido y distante, como por ejemplo la superficie de Marte.

Los conceptos de interesante y peligroso son ambiguos en la naturaleza, pero pueden ser aprendidos utilizando las redes neuronales, deacuerdo a la Dra. Ayanna Howard, ingeniera de investigación en robótica del JPL, especializada en inteligencia artificial creadora de tecnología inteligente para aplicaciones espaciales: «Podemos entrenar a un robot para que pueda saber si se encuentra en superficies rocosas y terreno peligroso, o superficie rocosas con características interesantes.» esto sin

duda entonces tiene un gran valor científico.

Las redes neuronales artificiales emulan el funcionamiento del cerebro humano, ya que emplean una gran red de elementos simples similares a las células cerebrales (neuronas), que tienen la capacidad de aprender a través de la presentación de ejemplos. Un robot que funciona con un sistema de este tipo aprende parecido a como lo hace un bebé o un niño pequeños.

Podemos decirle fácilmente a un robot que un cuadrado es un objeto equilátero con cuatro lados, pero la descripción de por ejemplo un perro no es tarea fácil. Sin embargo con las redes neuronales, podemos mostrar al robot muchos ejemplos de perros, y luego será capaz de reconocer a los perros en general. De manera similar, un modelo que emplee una red neuronal artificial puede aprender a clasificar el terreno si un experto le muestra imágenes de muchos tipos de terreno y les asocia una etiqueta de peligroso o seguro a cada uno, posteriormente cuando el robot vea un terreno en su ambiente de trabajo podrá determinar si este representa u peligroso o no.

2.3 Rovers trabajando

2.3.1 *Mars Exploration Rover*

La misión *Mars Exploration Rover* (MER) de la NASA consistió en enviar a los rovers Spirit y Opportunity para explorar el planeta en busca de fuentes de agua. Planificados para durar 90 días, los rovers excedieron las expectativas de todo el mundo, Spirit duró más de 7 años (20 veces más) y Opportunity duró 15 años (55 veces más), ambos devolvieron información valiosa sobre la composición geológica del planeta.

Para construir estos increíbles robots que funcionan con energía solar, el equipo del JPL utilizó FPGAt's Xilinx Virtex-4 tolerantes a la radiación, lo último en tecnología FPGA de grado espacial del momento; se utilizaron sobretodo para procesos cruciales de la fase de descenso y aterrizaje. Específicamente, se utilizaron FPGAt's XQVR4062 en cada módulo de aterrizaje para controlar las operaciones de pirotecnia, las cuales requieren tiempos de sincronización de mili segundos. Además, la NASA también usó FPGAt's XQVR1000 para construir el tablero de control de motores, el cual supervisa los motores de las ruedas, dirección, brazos, cámaras y diversos instrumentos.

2.3.2 *Mars Science Laboratory*

El siguiente rover en viajar a Marte fue *Curiosity*, a través del proyecto *Mars Science Laboratory* (MSL), se lanzó en 2011 y viajó durante ocho meses en un viaje de más de 50 millones kilómetros para una misión pensada para dos años. Diseñado para funcionar con energía nuclear, hoy en día se encuentra aún navegando en territorio marciano, su misión es tratar de determinar si el planeta alguna vez tuvo algún tipo de forma de vida microbiana. Después de ocho años de operación es sin duda uno de los sistemas mejor diseñados y seguramente continuará sorprendiéndonos.

Curiosity está equipado con un conjunto de doce cámaras de ingeniería, y cuenta con importantes sistemas de telecomunicaciones como el transmisor y receptor de banda X para comunicarse con la Tierra, así mismo integra una radio definido por software en banda UHF para comunicarse con los orbitadores de Marte que funcionan como ruta principal para el retorno de datos a la Tierra.

Los productos de grado espacial Xilinx permitieron la construcción de sistemas y instrumentos clave como el generador de imágenes y el procesador del MAHLI, que es una cámara colocada en el brazo robótico del rover, adquiere imágenes y hace la función equivalente de la lupa de mano de un geólogo. Así mismo, el sistema MALI procesa las imágenes de todas las cámaras integradas utilizando dispositivos FPGA Virtex-II (XQR2V3000) tolerantes a la radiación.

2.3.3 *Mars 2020*

No tiene mucho que arrancó la misión MARS 2020 con el nuevo rover explorador *Perseverance*, fue lanzado desde cabo cañaveral, florida el 30 de junio de 2020 a bordo del cohete Atlas V-541 y aterrizó con éxito en Marte el 18 de febrero de 2021 en el cráter Jezero. Su misión: nada menos que buscar signos de vida antigua y recolectar muestras de roca y suelo para un posible envió a la Tierra. Su tecnología está basada en la misión MSL (que tuvo buenos resultados). El rover, obtuvo su nombre mediante un concurso convocado por la NASA, donde miles de estudiantes propusieron nombres para bautizar al nuevo explorador, el ganador fue Alexander Mather, alumno de séptimo grado, quien propuso el nombre **Perseverance** en alusión a los nombres anteriores de los rovers que siempre evocan las mejores cualidades en los seres humanos: Curiosity, InSight, Spirit y Opportunity (Curiosidad, Visión, espíritu y oportunidad) y ahora Perseverance (perseverancia), que tiene cuatro objetivos específicos:

1. Buscar habitabilidad identificar entornos pasados capaces de sustentar vida microbiana.

2. Buscar signos de vida pasada microbiana (bio-firmas) en los ambientes habitables, particularmente en rocas especiales que se sabe preservan los signos de vida a lo largo del tiempo

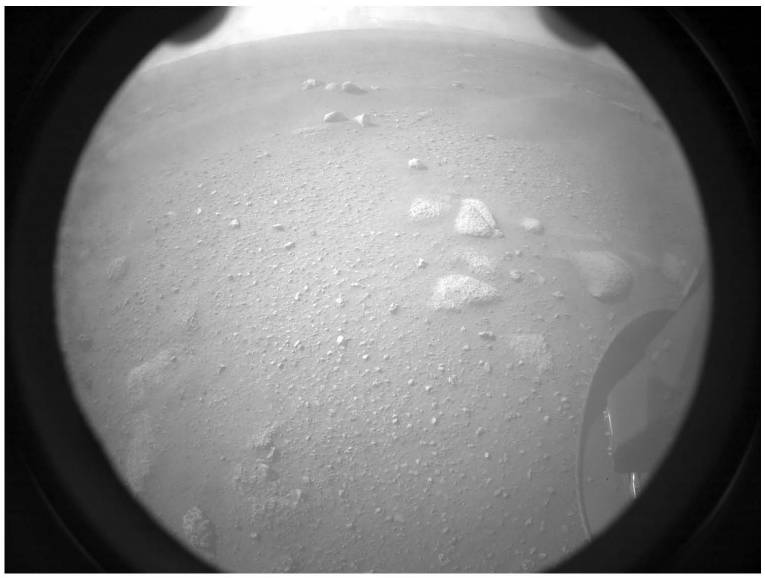
3. Recolecte muestras de roca y suelo y almacenarlas en la superficie marciana.

4. Probar la producción de oxígeno de la atmósfera marciana

No suficiente, sujeto al vientre de Perseverance viaja un pequeño helicóptero de demostración de tecnología (fig. 14 b), **Ingenuity** (ingenuidad) que pretende probar el primer vuelo propulsado en el Planeta Rojo.

(a) Perseverance

(b) Ingenuity



(c) Primera foto de Perseverance

Figura 14: Fotos de la misión Mars 2020[58]

3

CON LOS OJOS DE UN ROVER

En este capítulo se introduce la teoría del procesamiento de imágenes mediante visión artificial, se abordan algunos aspectos técnicos y terminología importante sobre las cámaras digitales. También, se resumen las principales características de las dos tecnologías para elaborar sensores ópticos. Por otro lado, se describen las cámaras a bordo de los rover exploradores de las misiones MER y MSL y el modo en que sus imágenes son procesadas.

Si uno echa un vistazo al árbol evolutivo de los animales y compara diferentes especímenes típicos en cada una de sus ramas, inevitablemente se llega a una conclusión que es evidente y extraordinaria: *¡los ojos han evolucionado muchas veces!, repetida e independientemente.*. Las células sensibles a la luz de todos los ojos de los seres vivos, evolucionaron a partir del sistema nervioso o mediante células cutáneas, solo se necesitaron esos dos tipos de células para producir ojos cristalinos y ojos facetarios, y también por supuesto, una gran cantidad de generaciones a lo largo de millones de años de evolución[20].

La visión es sin duda el sentido más avanzado que tenemos, por lo que no es de sorprender que las imágenes desempeñen un papel tan importante en la percepción humana; las primeras imágenes simbólicas surgieron como dibujos prehistóricos en paredes de cuevas o como tallados abstractos en monumentos de piedra, este comportamiento de representar la realidad se mantuvo siempre a lo largo de la historia del hombre, y se adaptó conforme a las mejoras tecnológicas de cada época, las cuales permitieron grabar imágenes con más realismo cada vez (pinturas y demás tipos de grabación indirecta)[1]. Sin embargo, el desarrollo de la fotografía revolucionó el acto en si mismo al permitir la grabación directa de imágenes, primero con la fotografía química a principios de 1800 y posteriormente con la fotografía electrónica, primero con sensores analógicos y más adelante con sensores digitales.

Dado que la visión es un sentido tan importante, el procesamiento de imágenes también se ha vuelto igual de importante. Las imágenes se pueden procesar para mejorar su contenido subjetivo y extraer información útil, con esto no solo se emula uno de los sentidos humanos más complejos y más abundantes en información, sino que también es posible aumentar y mejorar las capacidades de la visión humana para ver objetos muy distantes o en longitudes de onda invisibles para el ojo humano.

Ningún sensor de modalidad única puede proporcionar datos suficientes para extraer todas las características relevantes del entorno, pero la visión es la modalidad más rica en información. La visión es el principal modo sensorial para los explo-

1 Las imágenes grabadas de manera indirecta son aquellas donde el patrón de intensidad de luz no se usa directamente para producir la imagen.

radores planetarios que utilizan la observación a distancia para navegar y evitar obstáculos. Además, la visión es la forma más rica en información de datos sensoriales, en vehículos autónomos es el medio principal para generar mapas de la localidad que transitan. Además si se cuenta con una visión estéreo, se pueden reconocer objetos y determinar su posición y orientación relativa en el espacio[79].

3.1 Visión por computadora

La ciencia y la ingeniería para el procesamiento de imágenes y reconocimiento de patrones se ha desarrollado rápidamente el las últimas décadas, el procesamiento de imágenes digitales es un conjunto de técnicas aplicadas a la representación digital de una imagen que forma una escena, esto con el fin de reconocer o clasificar algunos elementos de interés y así facilitar un análisis posterior mediante un sistema de visión por computadora [15]. Las técnicas de procesamiento de imágenes se aplican cuando es necesario lo siguiente:

- Mejorar o modificar una imagen
- Destacar algún aspecto de la información contenida en ella
- Medir, comparar o clasificar un elemento en la imagen
- Reconocer el contenido de la imagen.

La visión por computadora tiene como objetivo hacer que las computadoras comprendan y procesen con precisión los datos visuales de manera eficiente, videos e imágenes; el objetivo principal de la visión por computadora es proporcionar a las computadoras el mismo tipo de capacidad y funcionalidad del cerebro humano en cuanto al sentido de la vista.

Teóricamente, la visión por computadora alude al control lógico que estudia cómo separar los datos de las imágenes en marcos artificiales. Los subdominios de la visión por computadora incluyen detección y reconocimiento de objetos, estimación de objetos, posición de objetos, detección de eventos, reconstrucción de escenas, restauración de imágenes, edición de imágenes, mejoramiento de video y aprendizaje estadístico.

El desarrollo exitoso de un sistema de visión depende del entendimiento de todas las partes que conforman la cadena de imagen. El atractivo y la complejidad de la visión artificial radica en el amplio rango de campos especializados en ingeniería que la conforman, algunos de ellos son:

- Ingeniería mecánica
- Ingeniería electrónica
- Ingeniería óptica
- Ingeniería de software

Contar con un modelo de visión por computadora permite definir los niveles y procesos que se realizan sobre una escena para llegar a una interpretación final de la imagen; la figura 15 muestra uno de estos modelos, se divide por tres bloques de procesamiento: nivel bajo, nivel intermedio y nivel alto. En el nivel bajo existen acciones sobre la imagen que definen a la escena, las cuales corresponden al suavizado, umbralización, eliminación de ruido, definición de bordes, análisis de textura, etc. En cambio, en el nivel intermedio se realizan acciones tales como definir limites, regiones y superficies que se relacionen con los *objetos* presentes en la imagen. Por último en el nivel alto, se entablan relaciones entre los objetos y así realizar la interpretación o descripción de la escena.

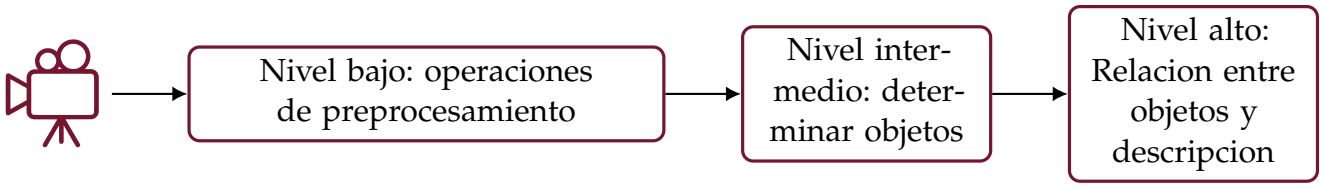


Figura 15: Modelo de visión por computadora

El primer requisito para capturar una imagen electrónica es contar con algún tipo de sensor para detectar y cuantificar la luz entrante[2]; según el sensor seleccionado y los demás componentes que conforman al sistema, se puede decir que las imágenes capturadas con dicho sistema son de buena o mala *calidad*, Así mismo, este nivel de calidad se determina en base a cuatro componentes principales: resolución, rango dinámico, reproducción de color/ profundidad de bits y rendimiento con poca luz. Se invita al lector a revisar el anexo A donde se definen conceptos importantes sobre visión artificial y procesamiento de imagen.

3.2 Procesamiento de imagen

El procesamiento de imagen consiste en someter una imagen a una serie de operaciones matemáticas para obtener un resultado deseado, este puede ser una mejora, la detección de alguna característica o evento crítico, la medida de un objeto o característica clave dentro de la imagen, la clasificación o detección de objetos dentro de la imagen, o bien la descripción de una escena. Existe un gran número de operaciones con las cuales se puede procesar una imagen, dependiendo de que información se quiera obtener de la escena se elegirá el proceso adecuado y así mismo la serie de pasos para obtener el resultado deseado; e procesamiento de imagen es tan amplio que se subdivide en varias ramas, algunas subdivisiones se clasifican incluso según la aplicación. A continuación se describen brevemente algunas de estas divisiones:

3.2.1 *Sensores para percepción visual*

La adquisición de imágenes o vídeo para su posterior almacenamiento o tratamiento en tiempo real, se realiza mediante cámaras analógicas (convencionales) y cá-

2 En la mayoría de las aplicaciones, también es necesario que el sensor sea direccional, de modo que responda principalmente a la luz que llega al sensor desde una dirección particular, sin esta sensibilidad de dirección, el sensor integrará efectivamente la luz que llega al sensor desde todas las direcciones.

maras digitales, las cámaras convencionales son aquellas que utilizan una película sensible a la luz y mediante tratamiento químico se revela y se obtiene la imagen. Por otro lado, la cámara digital es aquella en la cual la imagen obtenida queda guardada en alguna clase de almacenamiento digital o memoria electrónica.

En la figura 16 se muestra el diagrama de un sistema para la adquisición de imágenes digitales. El sensor convierte la luz reflejada en la escena en señales eléctricas[3], el controlador utiliza un software con el cual se comunica con la cámara digital y le permite manejar los datos a voluntad, con esto las imágenes pueden ser almacenadas en memoria como valores binarios y posteriormente desplegaras o procesarlas.

Como se mencionó anteriormente, existen dos tecnología diferentes para capturar imágenes de forma digital: los sensores CCD y los CMOS, los primeros ya llevan bastante tiempo en el mercado y los segundos se vienen usando ampliamente en cámaras más modernas.

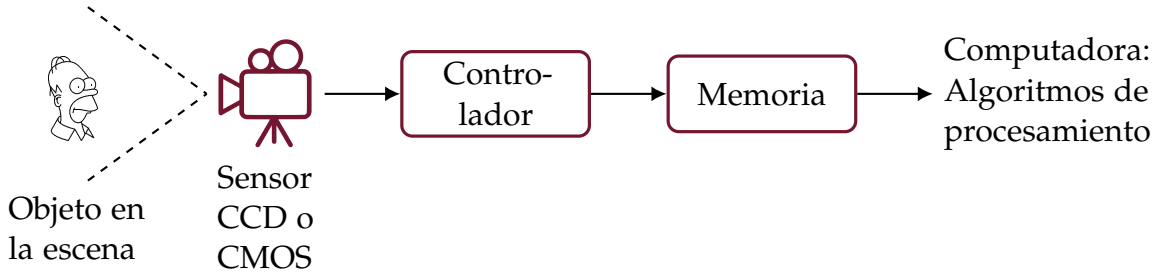


Figura 16: Sistema para la adquisición de imágenes digitales

3.2.2 *Sensor de luz CCD*

El primer dispositivo electrónico más difundido en cámaras para la adquisición de imágenes por digitales es el arreglo CCD, aunque es una tecnología antigua aun se sigue utilizando debido a que dota de ciertos atributos a las cámaras que la utilizan, entre ellos están:

- Alta resolución
- Son fáciles de operar
- Dimensiones pequeñas
- Bajo consumo de potencia
- Durabilidad
- Bajo costo

Un sensor CCD es un circuito integrado sensible a la luz diseñado para convertir una imagen visual en una señal eléctrica, los lentes ópticos enfocan la escena sobre un substrato fotoconductivo (silicio), el dispositivo absorbe la luz y la almacena

3 si son analógicas se convierten a binarias o digitales mediante un convertidor analógico digital

como una carga en miles de pequeños capacitores rectangulares, el silicio forma una sola capa común para todos los capacitores y las otras placas son pequeños electrodos de metal individuales separados del silicio fotoconductor por una fina película aislante de dióxido de silicio (SiO_2); tal como se observa en la figura 17, la luz que incide sobre el sustrato hace que los capacitores se carguen y descarguen a un nivel proporcional a la intensidad de la luz, incidiendo directamente bajo las pequeñas placas capacitivas.

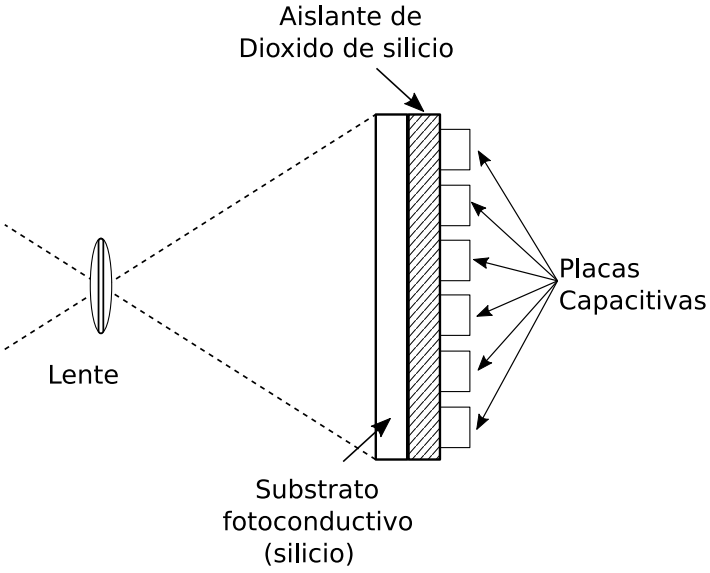


Figura 17: Corte transversal de un dispositivo CCD [10].

En los CCD, las pequeñas celdas capacitivas están acomodadas en un arreglo rectangular o cuadrado, las dimensiones estándar para estos arreglos son: 640×480, 768×576, 512×512, 1024×1024, o mayores, De modo que, a mayor número de celdas mayor es la resolución lograda de la escena y una alta resolución significa mejor definición en los detalles.

Para que el sensor pueda leer una imagen de la escena, se deben aplicar voltajes a los electrodos de los capacitores en una secuencia de derecha a izquierda, cuando se aplica el voltaje de control, el capacitor se descarga y una señal analógica se transfiere al substrato de silicio común a todos ellos.

En los sensores CCD, la carga de cada píxel se transfiere a través de un número muy limitado de nodos de salida (a menudo solo uno) para convertirse en voltaje y almacenarse en un búfer para enviarse fuera del chip como una señal analógica.

Así mismo, existen dos tipos de sensores CCD: para captura de linea y para captura de área, estos últimos son preferidos para capturar imágenes. los CCD incluyen la circuitería que almacena y transfiere su carga a un registro de desplazamiento el cual convierte el arreglo espacial de cargas en el CCD a una señal analógica de video, como se muestra en la figura 18.

En la señal de video van combinadas tanto la información de temporizado para indicar la posición vertical y horizontal del sensor así como su valor; las señales de temporizado o sincronía deben incluir el pulso de sincronía vertical (VSYNC)

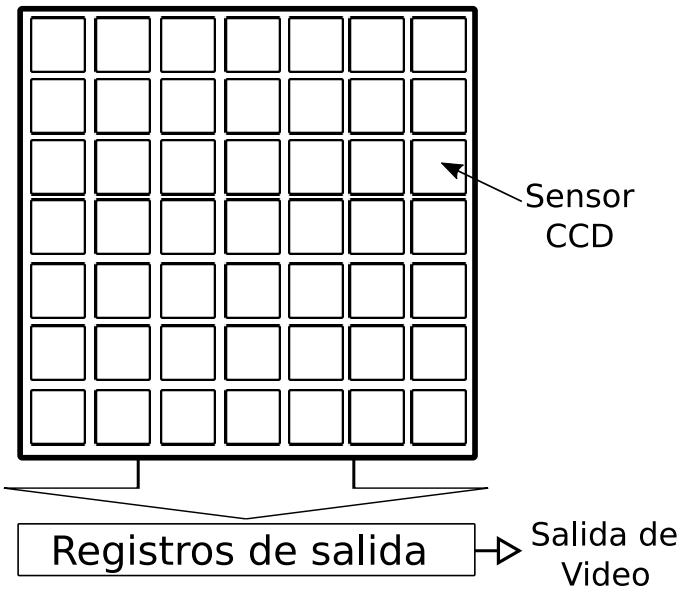


Figura 18: Arreglo de sensores CCD [10].

para identificar el principio de un campo (field) y el pulso de sincronía horizontal (HSYNC) para indicar el principio de una linea (line), de modo que dos campos entrelazados forman un cuadro (frame).

El temporizado de estas señales puede variar para los diferentes formatos de video existentes, pero el software de configuración de las cámaras del controlador se encarga de seleccionar el temporizado adecuado para cada cámara.

3.2.3 *Sensor de luz CMOS*

Los sensores CMOS también llamados sensores de de píxeles activos **APS, por sus siglas en inglés!** han desplazado al los CCD en muchas aplicaciones por su flexibilidad y menor costo; su principio de funcionamiento es similar a los CCD en cuanto a que existe un arreglo bidimensional de sensores CMOS que adquieren la luz incidente en ellos, no obstante la diferencia de los dos sensores radica principalmente en su tecnología de fabricación.

En cada pixel de los sensores CMOS hay varios transistores para amplificar y desplazar la carga a través de conductores muy simples. Además, es posible leer individualmente cada pixel, lo cual hace que su uso sea muy flexible, esto quiere decir que cada píxel tiene su propia conversión de carga a voltaje de modo que el chip entrega bits digitales.

Existen diferentes diseños de sensores CMOS con variantes en la topología de sus transistores, un ejemplo se muestra en la figura 19, en este ejemplo la estructura del sensor CMOS tiene tres transistores y un fotodiodo, el transistor RESET (T_{RST}) tiene la función de dejar cargar o descargar el sensor, el *transistor seguidor de fuente* (T_{SF}, source follower) sostiene el voltaje logrado por el pixel y por último el transistor selector (T_{SEL}) permite la lectura de una fila del arreglo de píxeles por la electrónica

externa al arreglo.

Por lo tanto, cada pixel cuenta con su conversor de carga a voltaje y también es probable que además contenga amplificadores, circuitos para corrección de ruido y circuitos para digitalización. No obstante, los circuitos adicionales en el píxeles pueden reducen la sensibilidad del fotodiodo, ya que el impacto de los fotones no es el mismo en la totalidad del sensor.

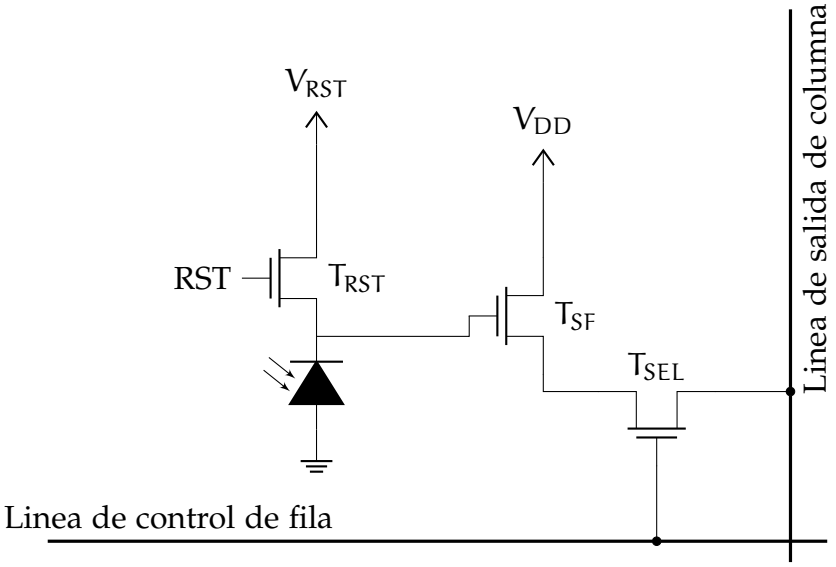


Figura 19: Un sensor de pixel activo (APS) CMOS con tres transistores [10].

3.2.4 *Comparación de sensores CCD y CMOS*

Aunque existen diferencias tecnológicas entre los sensores CCD y CMOS, la selección de ellos depende de la aplicación y del fabricante. En la actualidad, existen fabricantes que ofrecen soluciones con ambos tipos de tecnología. En la tabla 4, se presenta la comparación de estos dos tipos de tecnologías.
La ventaja de rendimiento de los generadores de imágenes CMOS sobre los CCD en términos de visión artificial merece una breve análisis: para la visión artificial, los parámetros clave son la velocidad y el menor ruido, los generadores de imágenes CMOS y CCD difieren en la forma en que las señales se convierten desde la carga de la señal a una señal analógica y finalmente a una señal digital. En el área CMOS y en los generadores de imágenes de exploración de línea, el extremo frontal de la ruta que siguen los datos es masivamente paralela, esto permite que cada amplificador tenga un ancho de banda bajo, para cuando la señal alcanza el cuello de botella del la ruta de datos, que normalmente es la interfaz entre el generador de imágenes y el circuito fuera del chip, los datos CMOS están ya en el dominio digital. Por el contrario, los CCD aunque poseen una alta velocidad y tienen una gran cantidad de canales de salida paralelos, pero no tan paralelos a los de las imágenes CMOS de alta velocidad. Por lo tanto, cada amplificador CCD tiene mayor ancho de banda, y esto resulta en mayor ruido.

Tabla 4: Comparación de características entre sensores CCD y CMOS[10].

Características	CCD	CMOS
Señal de salida del pixel	Carga de electrones	Voltaje
Señal de salida del dispositivo	Voltaje (analógico)	Bits (digitales)
Señal de salida de la cámara	Bits (digitales)	Bits (digitales)
Sensibilidad a la luz	Alta	Moderada
Ruido del sistema	Bajo	Moderado
Complejidad del sistema	Alta	Moderada
Complejidad del sensor	Baja	Alta
Capacidad de respuesta	Moderada	Ligeramente mejor
Rango dinámico	Alto	Moderado
Uniformidad	Alta	De baja a moderada
Obturación uniforme	Rápida	Pobre
Velocidad	Moderada a alta	Alta
Ventaneo	Limitado	Extensivo
Consumo de potencia	Alto	Bajo

La mayoría de los procesos de fabricación para generar imágenes con sensores CMOS, están ajustados para aplicaciones de alto volumen que solo muestran imágenes en el espectro visible. Estas imágenes no son muy sensibles al infrarrojo cercano[4] (NIR). Para obtener imágenes en el infrarrojo cercano (700 a 1000 nm), los generadores de imágenes deben tener una región de absorción de fotones más gruesa, esto se debe a que los fotones infrarrojos se absorben a mayor profundidad en el silicio que los fotones de luz visible. No obstante, el aumento del grosor de la capa epitaxial o epi[5] para mejorar la sensibilidad infrarroja, puede degradar la capacidad del generador de imágenes.

Por otro lado, los CCD se pueden fabricar con capas epi más gruesas al tiempo que conservan su capacidad. En algunos CCD de infrarrojo cercano, el epi tiene un grosor de más de 100 micras, en comparación con el epi de 5 a 10 micras en la mayoría de las cámaras CMOS. Por lo tanto, los CCD están diseñados específicamente para ser altamente sensibles en el infrarrojo cercano y son mucho más sensibles que los generadores de imágenes CMOS.

En cuanto a los diseños de sensores personalizadas, a menos de que el cambio sea menor, generalmente es más barato desarrollar un CCD personalizado que desarrollar un generador de imágenes CMOS personalizado, esto es porque en el desarrollo de los CMOS se utilizan máscaras submicrométricas profundas más costosas y también hay muchos más circuitos para diseñar en un dispositivo CMOS.

Como se mencionó anteriormente la selección del sensor adecuado depende de las tareas a realizar y de la aplicación, los generadores de imágenes de escaneo de área y línea CMOS superan a los CCD en la mayoría de las aplicaciones de imágenes

4 De hecho, están diseñados para ser lo más *insensible* posible en el NIR
5 El término epitaxia viene del griego epi (επι), que significa por encima y taxis (ταξισ), que significa de manera ordenada

en el rango visible. Sin embargo, los CCD de integración y retardo de tiempo (TDI, por sus siglas en inglés), utilizados para aplicaciones de alta velocidad y bajo nivel de luz, superan a los CMOS-TDI. La necesidad de obtener imágenes en el NIR puede hacer que los CCD sean una mejor opción para algunas aplicaciones de escaneo de área y línea. También, la necesidad de un ruido muy bajo introduce nuevas restricciones, ya que CMOS generalmente sigue superando a los CCD a altas velocidades de lectura. Así mismo en términos económicos, la ganancia precio-rendimiento también puede favorecer a los generadores de imágenes CCD o CMOS dependiendo del apalancamiento, el volumen y la seguridad del suministro.

ADQUISICIÓN Y FORMACIÓN DE LA IMAGEN: Los sensores entregan señales analógicas o digitales que contienen la información de la imagen o escena, sin embargo la imagen como tal se forma almacenando el conjunto de muestras o pixeles en algún tipo de memoria, el conjunto de dispositivos electrónicos utilizados para el proceso de adquisición o captura de la imagen o cuadro se conoce como grabador de fotogramas (FG, por sus siglas en inglés) y puede tener características diferentes según el sensor que se utilice y la electrónica involucrada. El FG hace el muestreo electrónico (o digitalización) de las señales de video entregadas por el sensor, o bien solo su captura. Convierte una imagen en un arreglo o puntos (pixeles) de datos digitales.

VISIÓN EMBEBIDA: La visión embebida (embedded visión) es el uso de la visión por computadora en sistemas embebidos[6]. Históricamente el procesamiento digital de imágenes comenzó y se expandió en aplicaciones industriales y científicas con algoritmos de procesamiento de imagen implementados en computadoras poderosas, hoy en día se tienen procesadores, de bajo costo, eficientes y muy poderosos, para aplicaciones de procesamiento de imagen en pequeños dispositivos móviles sin la necesidad de recurrir a una PC. Típicamente, los sistemas embebidos son dispositivos de computo pequeños y ligeros que pueden ser instalados en sistemas más grandes como un auto o un robot; la capacidad de capturar imágenes dentro de un solo sistema embebido permite diversos grados de autonomía al permitir que los sistemas mecánicos interactúen con el mundo que los rodea.

VISIÓN ESTÉREO: La estereopsis es el proceso donde la distancia (profundidad) es reconstruida a partir de dos imágenes ligeramente diferentes, la distancia al objeto puede ser calculada mediante los pixeles de cada imagen y los parámetros de la cámara; la visión binocular es ligeramente dispar de manera horizontal, permitiendo que la profundidad pueda ser calculada. El uso de una visión binocular estereoscópica proveniente de dos cámaras colocadas de diferente manera permite la estimación de rango 3D, las características son extraídas de las dos imágenes superpuestas una con otra, y la disparidad se calcula mediante las diferencias posicionales y características coincidentes.

6 Un sistema embebido (integrado) es un sistema de computación diseñado para realizar una o algunas pocas funciones dedicadas, la mayoría de los componentes se encuentran incluidos en la placa base.

3.3 Ver como un Rover

Los robots que exploran la superficie de Marte están equipados con un conjunto de cámaras científicas y de ingeniería, el objetivo principal de estas cámaras, es servir de apoyo durante la operaciones en la superficie marciana, incluso antes del aterrizaje en Marte (son los ojos del rover). Durante la conducción, los rover capturan imágenes delante y detrás de ellos para observar el terreno local y así poder determinar y evitar peligros de manera autónoma.

Las funciones de las cámaras sirven para tres funciones principales:

1. Apoyar la navegación durante la travesía
2. Respaldar el despliegue de dispositivos de adquisición de muestras e instrumentos científicos
3. Realizar observaciones científicas generales

De hecho, por las razones que se explicaron más temprano en este capítulo, las cámaras son el principal instrumento a bordo de los rover, dichas cámaras a menudo se montan en mástiles telescópicos con capacidades de movimiento de inclinación horizontal (un tipo de manipulador desde una perspectiva de control). Por ejemplo, el lander Viking llevó un par de cámaras estereoscópicas montadas a 1 [m] de distancia, cada una en un mástil por separado y a 1.3 [m] por encima del suelo.

Todos los rovers de las misiones MER y MSL llevan un sistema de cámaras multiespectrales, panorámico y estereoscópico (PanCam), generalmente montado en un mástil de 1.5 a 2 [m] de altura idealmente[7]. El sistema PanCam cumple la primer función al generar una vista panorámica de 360ř del sitio de aterrizaje. Las imágenes en color son esenciales ya que el color proporciona una buena aproximación de la reflectancia y permite la diferenciación entre obstáculos y sombras. La detección de cambios en el albedo proporciona la base de la teoría **retinex** de percepción del color, según la cual el color se estima en una amplia variedad de intensidades mediante el uso de la reflectancia de la superficie en diferentes distribuciones de longitud de onda. Por lo tanto, la imagen a color es mejor para una percepción precisa del terreno, los escáneres multiespectrales estiman el color de la superficie al recuperar el albedo de forma independiente en tres bandas espectrales de color primario, por lo que el filtro fotográfico estándar suele ser RGB, que se extiende a lo largo de regímenes visibles y casi UV que ofrecen respuestas máximas de rojo, verde y azul y para longitudes de onda ultravioleta.

La mayoría de los sistemas de visión móvil para los rovers se han basado en tecnologías CCD de estado sólido, pese a que las tecnologías basada en CMOS están cada vez más disponibles, ya que los arreglos CCD son rápidos, robustos, con alta fidelidad geométrica y buena resolución. También para los módulos de aterrizaje planetario, se ha utilizado tradicionalmente, el escaneo de trama línea por línea

7 Un mástil de 1 [m] de altura da una visibilidad al horizonte local de 1.8 [km], mientras que uno de 2 [m] proporciona un horizonte a 3.5 [km] de distancia (en Marte).

para la construcción de la imagen, pero la movilidad introducida por los rovers planetarios restringe esto, las tareas de visión en tiempo real más exigentes requieren cuadros con velocidades de 2530 Hz (25 Hz para el estándar europeo CCIR y 30 Hz para el estándar EIA americano), pero esto requiere un hardware dedicado para el capturador de cuadros de alto rendimiento y que actualmente no es factible. Sin embargo, esto se puede relajar considerablemente para los sistemas de imágenes de vehículos móviles planetarios debido a la velocidad lenta de los vehículos y a las limitaciones de los recursos de procesamiento computacional a bordo.

3.3.1 *Las Cámaras del MER y MSL*

Los vehículos de la misión Mars Exploration Rovers, poseían diecisiete cámaras, algunas de ellas para la navegación del móvil, y otras para realizar investigaciones científicas[52]. Cada cámara contiene un arreglo óptico específico para su aplicación y se pueden clasificar de la siguiente forma:

- Cámaras de descenso
- Cámaras científicas
- Cámaras de ingeniería

3.3.1.1 *Cámaras de descenso*

Este sistema se utiliza para detectar el movimiento de la nave espacial y ajustarlo, utilizando retro-cohetes si es necesario. El Mars Science Laboratory, tenía un sistema visual aún más capaz: MARDI (Mars Descent Imager), quien proporcionó video de cuatro cuadros por segundo a alta resolución durante el aterrizaje del rover Curiosity, las imágenes eran a *color verdadero*, es decir, como lo vería el ojo humano. Además del sorprendente video, los datos que recopiló la cámara MARDI permitió a los científicos e ingenieros observar los procesos geológicos en una variedad de escalas, muestrear el perfil del viento horizontal, y crear mapas geológicos, geomórficos y transversales muy detallados del sitio de aterrizaje.

3.3.1.2 *Cámaras Científicas*

CÁMARAS MASTCAM: Cada rover tiene cuatro cámaras científicas Mastcam, que toma imágenes en color, imágenes estéreo tridimensionales y secuencias de video en color del terreno marciano, también tiene un zoom potente. El sistema consta de dos conjuntos de cámaras redundantes montados en un mástil que se extiende hacia arriba desde la plataforma de rover. Las cámaras funcionan de manera muy parecida a los ojos humanos, produciendo imágenes estéreo tridimensionales combinando dos imágenes de lado a lado, tomadas desde posiciones ligeramente diferentes.

CÁMARA CHEMCAM: Un par de cámaras que disparan un láser y analizan la composición elemental de materiales vaporizados en un área de menos de 1 [mm] en la superficie de rocas y suelos marcianos. La ChemCam también toma imágenes en escala de grises.

CÁMARA MAHLI: Es el equivalente de la lente de la mano de un geólogo y proporciona vistas de cerca de los minerales, texturas y estructuras en las rocas marcianas y la capa superficial de escombros rocosos y polvo. Con este dispositivo, los geólogos terrestres pueden ver características marcianas más pequeñas que el diámetro de un cabello humano.

3.3.1.3 *Cámaras de ingeniería*

CÁMARAS DE PREVENCIÓN DE OBSTÁCULOS (HAZCAMS): Son cuatro pares de cámaras de ingeniería para evitar peligros (Hazcams), montadas en la parte inferior de la parte delantera y trasera del móvil, estas cámaras en blanco y negro usan luz visible para capturar imágenes en 3D. Estas imágenes protegen al rover de perderse o chocar con obstáculos inesperados, funcionan en conjunto con el software que le permite al rover tomar sus propias decisiones de seguridad y pensar por sí mismo.

También, el rover utiliza estos pares de imágenes para trazar la forma del terreno hasta 3 metros delante de él. Las cámaras Hazcam utilizan un arreglo óptico de ojo de pez para ver lejos a ambos lados, ya que a diferencia de los ojos humanos, ellas no pueden moverse independientemente al estar montadas directamente en el cuerpo del vehículo.

Tabla 5: Especificaciones funcionales de la cámara HazCam. [53][57]

Parámetro	Descripción
Función principal	Ayuda en la navegación autónoma y evitación de obstáculos.
Ubicación	En la parte delantera y trasera del cuerpo, apuntando hacia el suelo, a 68 cm sobre el suelo; frente: a 16.6 cm entre el centro de los ojos izquierdo y derecho; espalda: 10 cm, a 78 cm sobre el suelo
Peso	Aproximadamente 9 onzas (250 gramos) cada una
Escala de grises	Longitudes de onda rojas centradas en 650 nanómetros
Tamaño de la imagen	1024 X 1024 píxeles
Resolución	2,1 mrad por píxel
Longitud focal	En foco alrededor de 10 cm hasta el infinito
Relación focal y campo de visión	Lente ojo de pez con cuadrado ND124°
Distancia estereoscópica	Estéreo que se extiende inmediatamente delante del móvil (distancia interocular de 10 cm para Hazcams traseros, 16 cm para Hazcams frontales) con una precisión de ±5 mm.
Otro	Cada uno tiene una cubierta de lente extraíble por única vez para proteger del polvo levantado al aterrizar

CÁMARAS DE NAVEGACIÓN (NAVCAMS): Son dos los pares de cámaras de navegación montadas en el mástil, estas cámaras en blanco y negro usan la luz visible para obtener imágenes panorámicas tridimensionales (3D). La unidad de las cámaras de navegación esta formado por un par de cámaras estéreo, cada una con un campo de visión de 45ř y permite la planeación en cuanto a la navegación terrestre por parte de los científicos e ingenieros de la tierra. Las NavCAms trabajan en con-

junto con las HazCams al proporcionar una vista complementaria del terreno.

Tabla 6: Especificaciones funcionales de la cámara NavCam. [53][57]

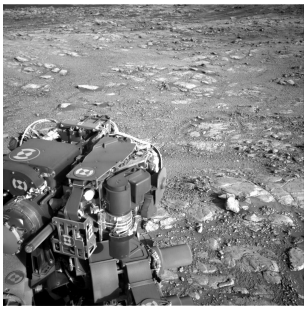
Parámetro	Descripción
Función principal	Ayuda en la navegación autónoma.
Ubicación	Montado en la parte delantera y trasera del cuerpo del rover, apuntando hacia el suelo; cada par de cámaras están separadas 42 cm
Peso	Aproximadamente 9 onzas (250 gramos) cada una
Escala de grises	Longitudes de onda rojas centradas en 650 nanómetros
Tamaño de la imagen	1024 X 1024 píxeles
Resolución	0.82 mrad por píxel
Longitud focal	en foco alrededor de 0.5 m hasta el infinito
Relación focal y campo de visión	Apertura fija f/12 y 45° cuadrado; el campo de visión es similar a una lente de 37 mm en una cámara de 35 mm
Distancia estereoscópica	Estéreo que se extiende a 100 metros (Distancia interocular de 42 cm)

Las cámaras de ingeniería en MSL se agrupan en dos conjuntos de seis cámaras cada uno: un conjunto de cámaras está conectado a una computadora móvil denominada A y el otro conjunto está conectado a la computadora móvil B. Las Navcams y Hazcams frontales proporcionan vistas similares desde cualquier computadora, no obstante los Hazcams traseros proporcionan diferentes vistas desde las dos computadoras debido a las diferentes ubicaciones de las Hazcams traseras A y B.

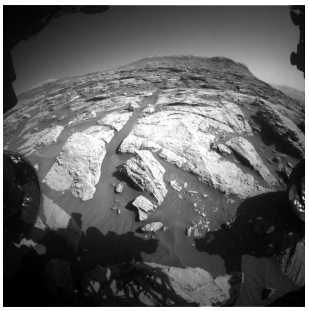
El rover Curiosity tiene cinco cámaras científicas, entre ellas las cámaras Mastcams que ven el mundo a color en dos resoluciones diferentes, la cámara MAHLI (pronunciada Molly) en la torreta al final del brazo, a color y con un lente gran angular que puede acercarse a un objetivo o realizar imágenes a distancia y la cámara MARDI que está fija al Cuerpo del rover, apuntando hacia abajo, teniendo una vista de la superficie por debajo del rover. Juntos, estos tres instrumentos a menudo se denominan cámaras *MMM*. Tienen un sensores, software y electrónica similares, y solo difieren en cuanto a sus arreglos ópticos.

En cuanto a los paramentos de procesamiento, posee un Procesador central endurecido contra radiación y con arquitectura PowerPC 750: BAE RAD 750. Funciona a una velocidad de hasta 200 [MHz], 10 veces la velocidad encontrada en las computadoras de los rover Spirit y Opportunity.

Curiosity tiene una memoria de 2 [GB] de memoria flash (8 veces más que en Spirit y Opportunity) y una memoria integrada de 256 [MB], ambas con detección y corrección de errores; finalmente su memoria EEPROM es de 256 KB (aproximadamente 8 veces con más capacidad que la de la misión MER).

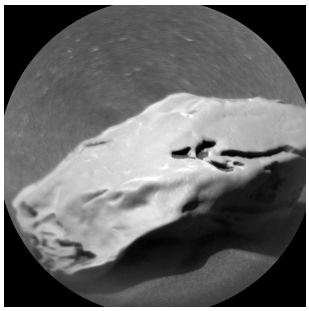


(a) NavCam



(b) HazCam

(c) MastCam



(d) ChemCam

Figura 20: Imagenes de las diferentes cámaras del rover Curiosity [56]

3.3.2 *Requerimientos de las cámaras MSL*

Las cámaras de ingeniería MSL fueron construidas en el JPL por el mismo grupo que construyó las cámaras MER. Las cámaras de ingeniería MSL están equipadas con calentadores un poco más potentes que las cámaras MER, lo que permite su uso potencial en condiciones más frías.

Cada cámara de ingeniería MSL se compone de dos carcasas mecánicas: un detector/cabezal óptico y una caja electrónica conectada mediante un cable flexible, estas se pueden observar en la Fig. 21, donde las cajas electrónicas tienen un tamaño aproximado de 67 Œ 69 Œ 34 mm. La Tabla 7 proporciona algunas de las propiedades más importantes de las cámaras de ingeniería MSL y la tabla 8 proporcionan un resumen del rendimiento y las características funcionales. En total se construyeron 26 cámaras para MSL (12 unidades de vuelo, 4 repuestos de vuelo y 10 unidades de ingeniería).

Los requerimientos para las cámaras de ingeniería MSL fueron heredados y adaptados a la misión MER. Aunque no hay requisitos científicos sobre las cámaras de ingeniería MSL, las cámaras devuelven imágenes que ayudan a cumplir los objetivos científicos de los instrumentos individuales al proporcionar un soporte contextual para muchas de las observaciones de la carga útil científica.

3.3.2.1 *Óptica*

Las propiedades ópticas de las cámaras de ingeniería se describen en detalle en [26], entre lo más relevante se puede decir que las cámaras Navcam usan lentes de longitud focal fija de 14,67 mm f/12 que proporcionan un campo de visión de 45

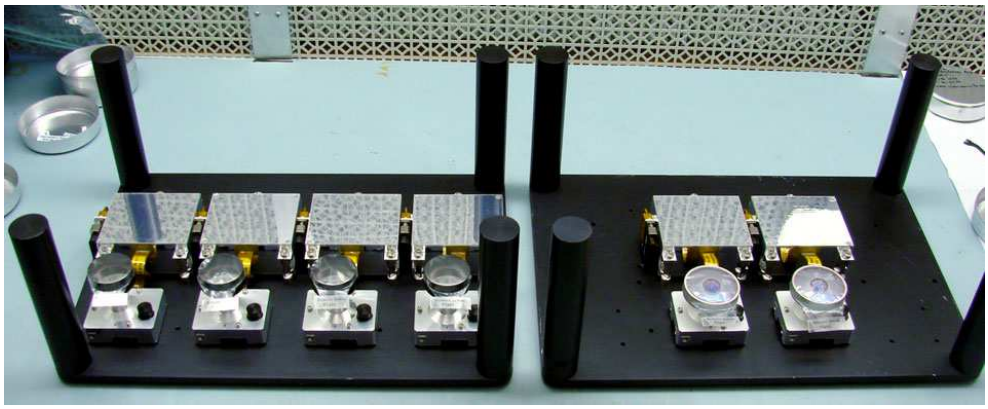


Figura 21: Cuatro cámaras de vuelo Navcams (izquierda) y dos cámaras de vuelo Hazcams (derecha). Las lentes de la cámara tienen una cubierta protectora que se quita antes del vuelo

Œ 45 grados (y diagonal de 60,7 grados) con una escala de píxeles, en el centro del campo de visión, de 0,82 mrad / píxel, así mismo, la profundidad de campo de la cámara Navcam varía de 0,5 metros a infinito, con una distancia hiperfocal de 1,0 metros.

Las cámaras Hazcam utilizan lentes de ojo de pez f-theta[8] de longitud focal fija de 5,58 mm f / 15 con un campo de visión de 124 Œ 124 grados y un FOV diagonal de 180 grados. La escala de píxeles en el centro de una imagen es de 2,1 mrad / píxel y la profundidad de campo varía de 0,10 metros a infinito, con una distancia hiperfocal de 0,5 metros. Tanto la cámara Navcam como la cámara Hazcam tienen filtros de paso de banda que cubren longitudes de inda desde 580 [nm] hasta aproximadamente 800 [nm].

Tabla 7: Propiedades ópticas de las cámaras de ingeniería MSL [53][26]

	NavCam	HazCam
Escala de píxeles en el centro del campo de visión	0.82 mrad/pixel	2.1 mrad/pixel
Longitud Focal	14.67 mm	5.58 mm
Número f	12	15
Diámetro de la pupila de entrada	1.25 mm	0.37 mm
Campo de visión (horizontal x vertical)	45 x 45 deg	124 x 124 deg
Campo de visión diagonal	67 deg	180 deg
Profundidad de campo	0.5 metros- infinito.	0.10 metros-infinito
Distancia hiperfocal	1.0 m	0.5 m
Rango espectral	600-800 nm	600-800 nm

3.3.2.2 *Detector*

Los detectores de imagen de las cámaras de ingeniería MSL se ensamblaron y empaquetaron usando obleas CCD, el detector es un dispositivo de transferencia

8 La terminología f-theta se refiere al mapeo de ángulos iguales en el espacio del objeto a distancias correspondientemente iguales en el plano de la imagen.

Tabla 8: Resumen de la configuración de las cámaras de ingeniería MSL [53][57]

Propiedad	NavCam	HazCam Frontal	HazCam trasera
Distancia interocular	42.4 cm	16.7 cm	10 cm
Diferencia de alineación estéreo	<1 deg	<2 deg	<2 deg
Dirección de apuntamiento Boresight	0-360 deg, acimut -87 a +91 deg, elevación	45 deg debajo del horizonte nominal	45 deg debajo del horizonte nominal
Altura sobre la superficie marciana	aprox 1.9 m	0.68 m	0.78 m
Masa (por cámara)	220 g	245 g	
Potencia (por cámara)	2.15 W		
Dimensiones	67 x 69 x 34 mm 41 x 51 x 15 mm	(electrónica) (cabeza detectora)	

de cuadros con una región de imagen de 12,3 mm x 12,3 mm que contiene 1024 x 1024 píxeles. El detector tiene 3 modos de lectura: fotograma completo, 4 x 1 en bandejas y el modo ventanas. El tiempo de lectura del detector es de 5.4 segundos para el modo de fotograma completo, 1.4 segundos para el modo agrupado y un tiempo de lectura variable (proporcional al tamaño de la ventana) de menos de 5.4 segundos para el modo de ventana. La capacidad total de un solo píxel es de aproximadamente 170,000 e-, la ganancia del detector es de 50 e-/DN y el ruido de lectura de la cadena de transferencia de señal es de aproximadamente 0.5 DN. Se puede encontrar una descripción más detallada de estos detectores en [52].

3.3.2.3 *Elemento de cálculo*

El software de vuelo para las imágenes MSL se ejecuta en la computadora principal del rover (RCE,por sus siglas en inglés), el móvil MSL contiene dos RCE funcionalmente idénticos: RCE A y RCE B; cada RCE está conectado a un conjunto dedicado de 6 cámaras de ingeniería (2 Navcams y 4 Hazcams), para un total de 12 cámaras, un RCE solo puede comunicarse con las seis cámaras que están conectadas a él; para adquirir imágenes del otro conjunto de 6 cámaras, el móvil debe conmutar al otro RCE. Durante las operaciones de superficie, un RCE se designará como la computadora principal, mientras que el otro RCE se apagará y se designará como una unidad de respaldo. El cambio de un RCE a otro probablemente se realizará solo si es necesario y se espera que ocurra con poca frecuencia durante la misión. Como los comandos de la cámara son idénticos para cualquiera de los RCE, los mismos comandos y secuencias de comandos se pueden cargar y ejecutar en cualquiera de los RCE, los parámetros específicos de una cámara en particular a bordo (por ejemplo, parámetros geométricos que describen cada una de las ópticas de las cámaras) deben cargarse en el RCE específico de interés y guardarse en la memoria no volátil del RCE.

3.3.2.4 *Software de vuelo*

El software de vuelo del MSL se heredaron y adaptaron a la misión MER, las capacidades de software a bordo del sistema MSL incluyen: exposición manual y

automática, almacenamiento de tabla de tiempo de exposición, escala de tiempo de exposición, generación de histogramas, suma de filas /columnas, generación de miniaturas, comparación de 12 a 8 bits, disminución de muestreo espacial, submarco espacial, sustracción del obturador , corrección de píxeles, corrección de viñeta de campo plano, gestión de modelos de cámara geométrica, procesamiento estéreo y colección de metadatos de imagen. El software de vuelo también utiliza el compresor de imagen *wavelet ICER* y el compresor de imagen *LOCO* para la compresión de una imagen sin pérdida. Las imágenes Hazcam y Navcam se obtienen mediante un comando de imagen autocontenido completamente especificado para la adquisición y el procesamiento de imágenes. Los panoramas de Navcam se obtienen ejecutando una serie de comandos de imagen individuales de forma secuencial. El software de vuelo enciende automáticamente las cámaras de interés en función de los comandos entrantes y apaga automáticamente las cámaras después de un período de inactividad. También se pueden alimentar hasta dos cámaras simultáneamente, las imágenes se leen y se almacenan en una tarjeta de memoria no volátil / interfaz de cámara (NVMCAM) antes de transferirlas al RCE para su posterior procesamiento.

3.3.3 *Innovación en Mars 2020*

Entre las inovaciones de la reciente misión Mars 2020, se incluye un nuevo acelerador de hardware basado en FPGA para el elemento de cálculo del sistema de visión, este será útil en la fase de aterrizaje y para la conducción autónoma. También Los FPGA Virtex-5QV endurecidos contra la radiación serán el procesador visual reprogramable en la tarjeta de visión artificial utilizada para acelerar ciertas tareas visuales como: rectificación, filtrado, detección y coincidencia de imágenes. También, se incluyen FPGAs en algunos de los instrumentos del Mastcam-Z, por ejemplo, un instrumento de imágenes estereoscópicas multiespectrales, que utiliza un FPGA Virtex-II tolerante a la radiación (XQR2V3000), así como para el escaner de entornos habitables Raman y el espectrómetro de luminiscencia para productos orgánicos y químicos (SHERLOC) que pertenecen a las cámaras pertenecientes al MAHLI. [71]

La tarjeta aceleradora de visión por computadora (CVAC, por sus siglas en inglés) acelera ciertas tareas de análisis estereoscópico y de odometría visual utilizando el FPGA Virtex-5QV. Implementa las porciones computacionalmente costosas de la tarea de análisis estéreo, incluidos los cálculos de rectificación de imagen, filtrado y disparidad. También, implementa partes de la tarea de odometría visual, incluido la detección y coincidencia de características que utiliza la detección de esquinas, y utiliza un algoritmo de suma de diferencias absolutas (SAD) aplicado sobre regiones locales entre imágenes en una secuencia. Finalmente, la CPU realiza los pasos restantes mediante encauzamiento. [47]

3.4 Aprendizaje profundo en rovers

Los sistemas de control autónomo a bordo de rovers planetarios y naves espaciales se benefician de poseer capacidades cognitivas tales como el aprendizaje, con

las cuales pueden adaptarse a situaciones inesperadas in situ. Sin embargo, los sistemas integrados a bordo de los rovers planetarios y naves espaciales rara vez implementan algoritmos de aprendizaje debido a las restricciones de potencia tamaño, y tasa de convergencia, así como los costos de los dispositivos que necesitan ser endurecidos contra la radiación.

Las misiones espaciales pueden beneficiarse de los algoritmos de aprendizaje automático para aumentar la autonomía operativa, las misiones a Marte o más lejanas se enfrentan a largas latencias que impiden la teleoperación. Por lo general, los algoritmos de aprendizaje automático son computacionalmente intensivos y requieren una potencia de procesamiento significativa, por lo que es una preocupación el ejecutar dichos algoritmos en rovers planetarios y demás hardware espacial[54].

4

LA CIENCIA DEL CEREBRO

Este capítulo está dedicado a las redes neuronales artificiales, su relación con la inteligencia artificial y el aprendizaje automático. Ademas, se incluye una breve recopilación histórica de su investigación hasta nuestros días y se describe su estructura, topologías básicas y modelos matemáticos, haciendo especial énfasis en su relación análoga con las redes neuronales biológicas, que sirvieron de inspiración para crear este concepto revolucionario. El capítulo finaliza describiendo el funcionamiento del clasificador neuronal *LIRA*, el cual es utilizado en este trabajo.

4.1 Máquinas inteligentes

Los problemas relacionados al pensamiento y la mente humana han atrapado el interés de los pensadores a lo largo de la historia. Desde tiempos antiguos, los filósofos, naturalistas y eruditos se han preocupado por estos problemas y se han planteado alguna de las siguientes preguntas: £Qué es la inteligencia?, £Cómo puede medirse la inteligencia, £Cómo funciona el cerebro?. Hoy en día, la mente humana es estudiada por biólogos, neurofisiólogos, psicólogos, psiquiatras y filósofos de diversas disciplinas[1], pero la respuestas a estas difíciles preguntas no han sido de todo convincentes. No obstante, en el proceso se han desarrollado muchas investigaciones en torno a la noción de inteligencia que han ayudado a distinguir ciertas características distintivas y algunas propiedades generales que presenta la inteligencia humana, por ejemplo la habilidad de enfrentar nuevas situaciones, la habilidad de resolver problemas, de responder preguntas y de elaborar planes. Sin embargo, aunque las preguntas anteriores son significativas cuando se trata de comprender mejor el funcionamiento del cerebro y la inteligencia, la pregunta central para el ingeniero es sobre cómo emular dicha inteligencia, es decir, £cómo crear una máquina inteligente que se comporte como una persona?

Existen algunos tipos de problemas que no pueden formularse mediante un algoritmo, problemas que dependen de muchos factores sutiles, problemas complejos que una computadora no podría calcular pero que el cerebro humano lo haría fácilmente. La diferencia es que los cerebros puede aprender, las computadoras no.

Si comparamos a las computadoras y el cerebro humano en términos de velocidad, notaremos, que en teoría, una computadora debería ser más poderosa, ya que estas contienen 10^9 unidades lógicas con un tiempo de conmutación de 10^{-9} segundos, mientras que el cerebro posee aproximadamente 10^{11} neuronas pero tan solo con un tiempo de conmutación de 10^{-3} segundos[36]. Sin embargo, se sabe que no es así, la estructura de las computadoras les permiten realizar operaciones a gran velocidad y almacenar datos de manera pasiva, pero carecen de la masiva estructura en paralelo del cerebro que le permite trabajar de manera continua, reflejando así

calidad y rendimiento por encima del tiempo de procesamiento. En la tabla 9, se muestra una comparación básica entre el cerebro y una computadora.

Tabla 9: Comparación entre el cerebro y una computadora [84].

	Cerebro	Computadora
Nř de unidades de conmutación	$\approx 10^{11}$	$\approx 10^{9}$
Unidad de procesamiento	Basada en neuronas	Basada en comp. lógicas
Tipo de cálculo	Masívamente paralelo	Generalmente en serie
Almacenamiento de datos	Asociativo	Basado en direcciones
Tiempo de conmutación	$\approx 10^{-3}$s	$\approx 10^{-9}$s
Operaciones de conmutación posibles	$\approx 10^{13}\frac{1}{s}$	$\approx 10^{18}\frac{1}{s}$
Operaciones de conmutación actuales	$\approx 10^{12}\frac{1}{s}$	$\approx 10^{10}\frac{1}{s}$

Hoy por hoy, gracias al esfuerzo de muchos investigadores y entusiastas del tema, existen dos campos de estudio relacionados a dotar a los sistemas y a las maquinas con cualidades de la inteligencia humana y otros seres pensantes, estos son: la *inteligencia artificial* y el *aprendizaje automático.*

INTELIGENCIA ARTIFICIAL: En 1958, El informático John McCarthy fue el responsable de introducir el término *inteligencia artificial* (IA), la cual es un área de estudio de las ciencias de la computación orientada a desarrollar sistemas capaces de participar en procesos de pensamiento similares a los humanos, tales como el pensamiento, el razonamiento y la auto corrección[33]. La IA siempre ha tenido como modelo las funcionalidades naturales del hombre enfocándose en distintos aspectos. Existen varios elementos que componen la ciencia de la inteligencia artificial, dentro de los cuales se pueden encontrar tres grandes ramas: lógica difusa, redes neuronales artificiales y algoritmos genéticos (ver figura 22) cada rama consta de características especiales así como de una aplicación especifica.

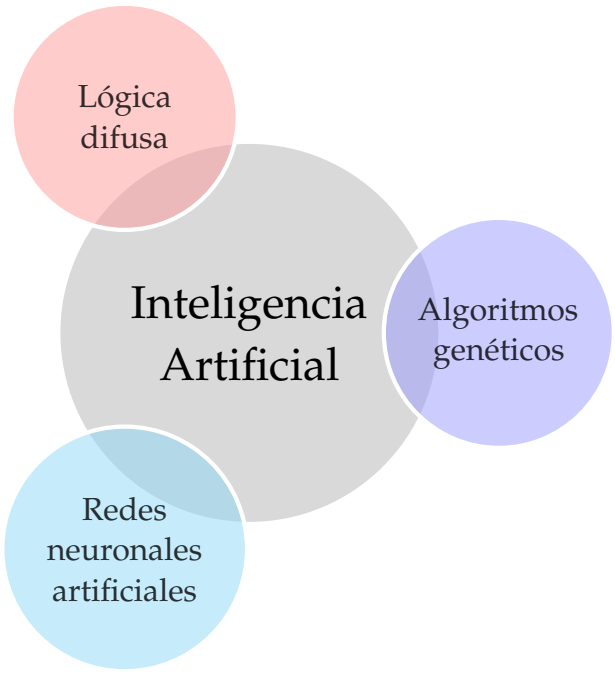


Figura 22: Mapa de los pilares que forman a la inteligencia artificial

APRENDIZAJE AUTOMÁTICO: El aprendizaje automático (machine learning, como se conoce en inglés) es un subcampo de las ciencias de la computación que estudia algoritmos para el procesamiento de datos, los cuales tienen la capacidad de aprender de los datos mismos. También, además de los algoritmos per se, el aprendizaje automático también incluye diferentes conjuntos de ideas que simplifican significativamente a otros algoritmos y que los hacen más adecuados para aplicaciones prácticas.

En el aprendizaje automático se tiene como finalidad obtener como resultado un modelo para una tarea dada, estos modelos pueden clasificarse de manera general en: modelos geométricos, modelos probabilistas o modelos lógicos; estos modelos pueden pertenecer a cualquier rama de las ciencias matemáticas o de las ciencias de la computación, incluyendo por supuesto la IA; si es adecuado para la tarea el aprendizaje automático, este puede tomarlo. En la figura 23 se muestra una imagen con algunos ejemplos de los modelos que pueden y son utilizados en el aprendizaje automático.

Para que los sistemas puedan *aprender* a realizar una tarea, es necesario *entrenar* a los algoritmos, en otras palabras, enseñarles qué es lo se deben hacer con los datos que procesan. Los algoritmos de aprendizaje automático no están diseñados para funcionar bien con un conjunto de datos en particular, sino para ejecutar una determinada tarea versátil. El entrenamiento o aprendizaje, como es correcto llamarlo, sirve para ajustar el algoritmo y tener un mejor rendimiento para el conjunto de datos de interés. Los tipos de aprendizaje utilizados son: supervisado, no supervisado, semi supervisado y por refuerzo.

4.2 Redes neuronales

Una de las ramas de la inteligencia artificial más utilizadas en sistemas que utilizan aprendizaje automático son las llamadas redes neuronales artificiales (ANN, por sus siglas en inglés). Las ANN son una familia de modelos computacionales que están inspirados en las redes neuronales biológicas, son utilizadas para estimar o aproximar funciones (generalmente desconocidas) que dependen de una gran cantidad de entradas. Por lo tanto, el estudio de las redes neuronales artificiales está motivado por su similitud con los sistemas biológicos que funcionan con gran éxito y que su principal enfoque es contar con células nerviosas simples, muy numerosas y que funcionan masivamente en paralelo. Además, uno sus aspectos más significativos es que tienen la capacidad de aprender.

Las redes contienen unidades de procesamiento simples pero numerosas.

Las ANN no necesitan ser programadas de manera explicita, aprenden de muestras de entrenamiento que les dan la capacidad de generalizar y asociar datos. Posterior a un entrenamiento exitoso, una red neuronal puede encontrar soluciones racionales para problemas similares o de la misma clase para los que no fueron entrenadas particularmente. Esto a su vez, resulta en un alto grado de tolerancia a fallas contra datos de entrada ruidosos [36].

Las redes neuronales son tolerantes a fallas.

Figura 23: Mapa de los modelos utilizados en aprendizaje automático

Los intentos por modelar el cerebro humano surgieron con la creación de la primera computadora, en un principio se comenzó desarrollando paradigmas de redes neuronales para dar solución a problemas relacionados con el procesamiento de sensores, reconocimiento de patrones, análisis de datos y control. Pero en las últimas décadas, las redes neuronales han pasado de la teoría a ofrecer verdaderas soluciones a problemas industriales y comerciales. La historia de las ANN comenzó a principios de la de década de 1940, casi simultáneamente con la historia de las computadoras electrónicas programables, este es un campo de investigación relativamente joven, tanto que varios de los pioneros aun siguen con vida, a continuación se presenta una breve recopilación de su historia con los aspectos más significativos.

4.2.1 *Breve historia de las redes neuronales*

A principio de la década de 1940, los trabajos desarrollados sobre la inteligencia artificial dividió la investigación sobre el pensamiento realizada hasta el momento en dos enfoques distintos: por un lado, el estudio de los procesos biológicos reales en el cerebro, y por otro, la obtención de modelos que emulen la fisiología de las redes neuronales biológicas. Uno de los artículos más sobresalientes fue el de Warren McCulloch y Walter Pitts en **1943** [80], este fue el primer intento por describir las funciones del sistema nervioso y el primer modelo conexionista, se introdujeron modelos de redes con las cuales se recreaban interruptores de umbral basados

en neuronas, demostrando así que incluso las redes simples de este tipo pueden calcular casi cualquier función lógica o aritmética.

Más tarde en **1949**, Donald O. Hebb formuló su famosa regla clásica de Hebb[27] que representa, en su forma más generalizada, la base de casi todos los procedimientos de aprendizaje neuronal; la regla implica que la conexión entre dos neuronas se fortalece cuando ambas neuronas están activas al mismo tiempo. Hebb trató de elaborar una teoría general comportamental y verificar su teoría, pero por la escasez de investigación neurologica de ese tiempo no pudo realizarlo.

A partir de **1951** comienza la edad de oro de las redes neuronales, Frank Rosenblatt inventó y propuso los perceptrones[66] (la arquitectura de las redes neuronales clásicas), Rosenblatt describió diferentes versiones del perceptrón, formuló y verificó su teorema de convergencia perceptrónica, describió las capas neuronales que imitan la retina, los interruptores de umbral y una regla de aprendizaje que ajusta los valores entre sus conexiones.

En **1961** Karl Steinbuch presentó desarrollos técnicos de la memoria asociativa, que pueden verse como predecesores de los recuerdos asociativos neuronales actuales[75]. Además, describió conceptos para técnicas neuronales y analizó sus posibilidades y límites.

Más tarde en su libro Learning Machines de **1965**, Nils Nilsson dio una visión general del progreso y los trabajos de el último período de investigación sobre las redes neuronales. En el suponía que los principios básicos del autoaprendizaje, y en general de los sistemas inteligentes, ya habían sido descubiertos. Hoy esta suposición parece ser una sobreestimación, pero en ese momento proporcionó una gran popularidad y suficientes fondos de investigación[36]. Sin embargo, en **1969** Marvin Minsky y Seymour Papert publicaron un análisis matemático preciso del perceptrón de Rosenblatt para demostrar que el modelo de perceptrón no era capaz de representar algunos problemas importantes (problema XOR y separabilidad lineal).

No obstante, análisis de Minsky y Papert aunado a que las computadoras en esa época no tenían suficientes capacidades para manejar redes de gran tamaño, hizo a muchos pensar que el fin de este paradigma estaba cerca; e en consecuencia otros métodos de inteligencia artificial comenzaron a ser desarrollados y tomaron el lugar de la investigación de las redes neuronales.

Esto puso fin a la popularidad y los fondos de investigación de redes neuronales en muchos lugares, sobretodo en los EUA donde resultó en una disminución casi completa en los fondos de investigación durante los siguientes 15 años. La investigación no cesó, pero no hubo conferencias ni otros eventos y por lo tanto solo se dieron unas pocas publicaciones. Sin embargo, el aislamiento de investigadores individuales hizo que aparecieran muchos paradigmas de redes neuronales desarrollados independientemente.

De **1969** a **1982** la investigación de las redes neuronales pasó a la clandestinidad en EUA, pero en la Unión Soviética (URSS), Europa y Japón esta investigación no se

detuvo.

Un ejemplo es el Instituto de Cibernética de la Academia de Ciencias en Kiev, Ucrania, que abrió un departamento de biocibernética en **1960** y cuyo fundador y supervisor durante mas de 30 años fue el doctor *Nicolay M. Amasov*.

En **1964** N. M. Amosov presentó una hipótesis que describía los mecanismos por los cuales los cerebros humanos procesan la información, esta fue publicada más tarde en **1965** como un tratado llamado: *Modelado del pensamiento y la mente* [1], el cual se convirtió en la Biblia para varias generaciones de investigadores de esta área. Según el propio científico, en cibernética todo se trata del modelado: modelando de células, de organismos, mente, sociedad. Las ideas que Amosov propuso en su libro fueron desarrolladas aún más en sus trabajos posteriores.

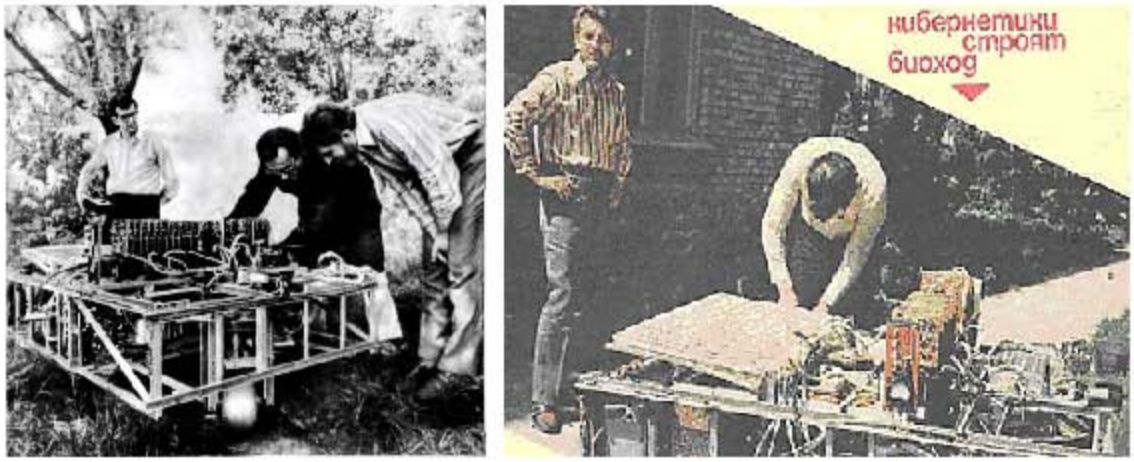
Los desarrollos de cibernética neuronal fueron estimulados por el trabajo dirigido a obtener resultados aplicables y que tuvieran un valor práctico. Para el departamento de Amosov esta investigación resultó estar naturalmente conectada a la creación de prototipos de robots móviles autónomos y al desarrollo de sistemas de control mediante redes neuronales. Cabe mencionar que en esa época comenzaba un área temática robottécnica que se hizo bastante popular en la URSS, incluso se dieron instrucciones directas del partido comunista y del gobierno para el desarrollo en esta área.

Figura 24: Estampa de Amosov, Ucrania[29]

Al principio el Dr. Amosov se opuso fuertemente, pero posteriormente se tranquilizó y como resultado desarrolló y estudió toda una familia de robots de este tipo. El Dr. *Ernst M. Kussul*, un miembro e investigador del departamento, inició y supervisó esta línea de investigación, además fue él quien dirigió el departamento de biocibernética después que Asomov dejara el puesto. De **1972** a **1975**, se creó el primer robot de transporte autónomo de la URSS, llamado *TAIR* (ver figura 25). El robot demostró tener movimientos intencionados en un entorno natural mediante la evasión de obstáculos. TAIR consistía en una carretilla de tres ruedas equipada con un sistema de sensores que incluían un telémetro y sensores táctiles, fue controlado mediante una red neuronal implementada en hardware.[1]

Aunque el número de trabajos relacionados con las redes neuronales artificiales disminuyo durante un tiempo considerable, su popularidad resurgió más tarde gracias a la tecnología CMOS que condujo a una explosión en la velocidad computacional. Y también gracias a que en **1974** Paul Werbos[81], para su disertación en Harvard, desarrolló un procedimiento de aprendizaje llamado retropropagación

1 Los nodos de la red se implementaron mediante circuitos con transistores especiales y las conexiones entre los nodos mediante resistencias.



(a) E. Kussul (izquierda) y robot TAIR (b) E. Kussul (derecha) preparando el robot para la caminata

Figura 25: Robot de transporte autónomo *TAIR* de la URSS [29]

del error (backpropagation, por su nombre en inglés) [81], el cual permitió entrenar eficientemente grandes redes con múltiples capas y que actualmente tiene gran importancia.

También, en los años **80t's** una nueva ola de interés mundial surgió debido a la publicación de John Hopfield[44], un investigador en el campo de la biofísica, él describió la analogía entre el modelo de la red neuronal de Hebb y una cierta clase de sistemas físicos inspirados en las leyes del magnetismo, aunque no fueron utilizadas ampliamente en aplicaciones técnicas hizo que cientos de científicos e ingenieros calificados se unieran nuevamente a la investigación de redes neuronales.

Alrededor de **1986** el nuevo término *neurocomputadora* apareció. Muchas conferencias internacionales sobre redes neuronales, neuro-computación y neuro-computadoras tuvieron lugar alrededor del mundo. Se establecieron cientos de empresas dedicadas al desarrollo y producción de esta tecnología. Todas las esperanzas asociadas con los primeros proyectos sobre la creación de inteligencia artificial ahora se depositaron en los neuro-computadores, que básicamente fueron vistos como prototipos de un *cerebro artificial*.

La primer neurocomputadora soviética llamada *NIC* se creó de **1988** a **1989** y se implementó como un archivo adjunto a una computadora personal. Fue desarrollada en el departamento de sistemas de redes de procesamiento de información, del Instituto de Cibernética de Kiev, Ucrania, bajo la dirección de Ernst M. Kussul quien también propuso y analizó un nuevo paradigma de red neuronal que permitió la creación de estructuras muy similares a como se comportan las neuronas, estas estructuras se conocen ahora como redes neuronales asociativas-proyectivas.

Posteriormente, de **1991** a **1992** el equipo ucraniano-japonés crearon una nueva neurocomputadora que utilizó una base de elementos más avanzados. La neurocompuadora fue nombrada B-512, y E. Kussul junto con sus colaboradores y discípulos: Tetyana Baydik, Dmitrij Rachkovskij, Mikhail Kussul, y Sergei Artykutsa, nueva-

mente participaron en el desarrollo y obtuvieron buenos resultados junto con los investigadores japoneses de *WACOM*[44].

Figura 26: Computadora B-512[44]

La última versión de la B-512 fue desarrollada y probada en tareas de reconocimiento de imágenes, por ejemplo, en la tarea de reconocimiento de cifras escritas a mano. En las siguientes dos décadas, algunas de las limitaciones tecnologías de las redes neuronales han sido superadas en cierta medida, mediante trabajos que exploraron otras líneas de investigación, y por los avances en en campo de la implementación de redes neuronales en por ejemplo, unidades de procesamiento gráfico (GPU, por sus siglas en inglés), así como la aparición de las redes neuronales complejas y las redes neuronales profundas, que hicieron que el enfoque de las ANN volviera nuevamente a ser muy popular. Actualmente se han obtenido resultados impresionantes en el reconocimiento de voz o texto y también para el descubrimiento de nuevos medicamentos.

En la actualidad, ANN se han convertido en un modelo popular y útil para la clasificación, agrupamiento, reconocimiento de patrones y predicción en muchas disciplinas. El gran potencial de las ANN es el procesamiento de alta velocidad mediante su implementación en dispositivos de naturaleza paralela.

Hasta el año **2006** no se sabia realmente como entrenar redes neuronales que pudieran superaran los enfoques más tradicionales, lo que cambió en ese año fue el descubrimiento de técnicas de aprendizaje en las llamadas redes neuronales profundas. Este conjunto de técnicas se conocen actualmente ahora como aprendizaje profundo; en la actualidad se han desarrollado aún más,las redes neuronales profundas y el aprendizaje profundo han logrado un rendimiento sobresaliente en muchos problemas importantes en los campos de visión por computadora, reconocimiento de voz y el procesamiento del lenguaje natural, por lo que hoy en día están siendo implementados a gran escala por empresas como Google, Microsoft y Facebook.

A continuación se muestra en orden cronológico, una linea de tiempo con con los acontecimientos importantes, antes mencionados, en la historia de las redes neuronales.

LINEA DE TIEMPO 1: *Historia de las redes neuronales*

1940	Trabajos realizados sobre IA dividen la investigación del pensamiento en dos enfoques: los procesos biologicos y la fisiologia de las redes neuronales.
1943	Creación del primer modelo conexionista para describir las funciones del sistema nervioso y las ANN por McCulloch y Pitts.
1949	Formulación de la teoría general comportamental de Hebb y su postulado de aprendizaje.
1957	Frank Rosenblatt desarrolla los perceptrones, se crea la arquitectura de las redes neuronales clásicas
1964	Hipótesis de N.M Amosov sobre los mecanismos de procesamiento de información del cerebro.
1965	Nils Nilsson y su visión general del progreso de las ANN, inicio de del ultimo periodo de oro en la investigación de ANN.
1969-1982	Retraso en la investigación de redes neuronales en USA, sin embargo los trabajos continúan en la URSS, Europa y Japón.
1972-1975	Desarrollo del primer vehículo autónomo *TAIR* en la URSS, mediante implementación de redes neuronales en hardware.
1974	Paul Werbos desarrola el procedimiento conocido como retropropagación (Backpropagation)
1980	Publicaciones de John Hopfield y una nueva ola de interés en las redes neuronales
1986	Aparición de conferencias sobre ANN y neurocomputadoras alrededor del mundo.
1988-1989	Primer neurocomputadora (NIC) desarrollada en el instituto de cibernética de Kiev, Ucrania. Bajo la dirección de Erns M. Kussul.
1991-1992	Desarrollo de la computadora B-512 por el equipo ucraniano-japonés.
2006-2020	Uso de técnicas de apren. profundo con rendimiento sobresaliente.

4.3 El sistema nervioso

El sistema de procesamiento de información de los vertebrados, conocido como el sistema nervioso, es el centro de control del cuerpo y su red de comunicación, dirige las funciones de los órganos y los sistemas corporales; nos permite interpretar que ocurre a nuestro al rededor y nos ayuda a decidir o reaccionar ante cualquier estimulo ambiental mediante respuestas musculares o endocrinas.

El tejido nervioso es el conjunto de células especializadas que conforman el sistema nervioso y está constituido por grupos de células nerviosas o neurona, así como estructuras conductoras (nervios) que transmiten información en forma de impulsos mediante cambios electroquímicos. Un nervio es un tejido constituido por un haz de fibras nerviosas y un tipo de células especiales de soporte y protección conocidas como células de glía o neuroglia.[2]

El sistema nervioso se puede agrupar en dos categorías principales: la primera es el sistema nervioso centra (SNC), es el centro de control de todo el sistema, consiste en el encéfalo y la médula espinal; todas las sensaciones y cambios en nuestro ambiente son enviadas por receptores y órganos sensoriales hacia el SNC para que sean interpretados, y después si es necesario, dar una respuesta.

La segunda categoría es el sistema nervioso periférico (SNP), consiste en todos los nervios que conectan al encéfalo y a la médula espinal con receptores sensoriales, músculos y glándulas. A su vez el SNP se divide en dos subcategorias: el sistema periférico *aferente*, que consiste en neuronas sensoriales o aferentes que llevan información de receptores de la periferia del cuerpo hacia el encéfalo y la médula espinal, y el sistema periférico *eferente*, que consiste en neuronas motoras o eferentes que llevan información desde el encéfalo y la médula espinal hacia los músculos y glándulas. Un mapa conceptual de las divisiones del sistema nervioso antes mencionadas es mostrado en la figura 27.

4.3.1 *Redes neuronales biológicas*

Como se dijo anteriormente, las células que componen principalmente al sistema nervioso son las neuronas, cuya función principal es procesar y transmitir información mediante señales electroquímicas. El cerebro, el sistema de procesamiento de información más complejo que conocemos, contiene poco más de 100 mil millones de ellas, cada una de las cuales posee en promedio más de 10 mil conexiones, las neuronas se pueden clasificar tanto por su función como por su estructura como: unipolares,bipolares y multipolares.

La figura 28 ofrece una representación esquemática de las partes principales que conforman a las neuronas; aunque su forma sea diferente cabe destacar que todas contienen tres partes fundamentales: un cuerpo (soma), un árbol de entradas (den-

2 Neuroglia significa pegamento nervioso, cerca del 60 % de las células del cerebro son células de neuroglia.

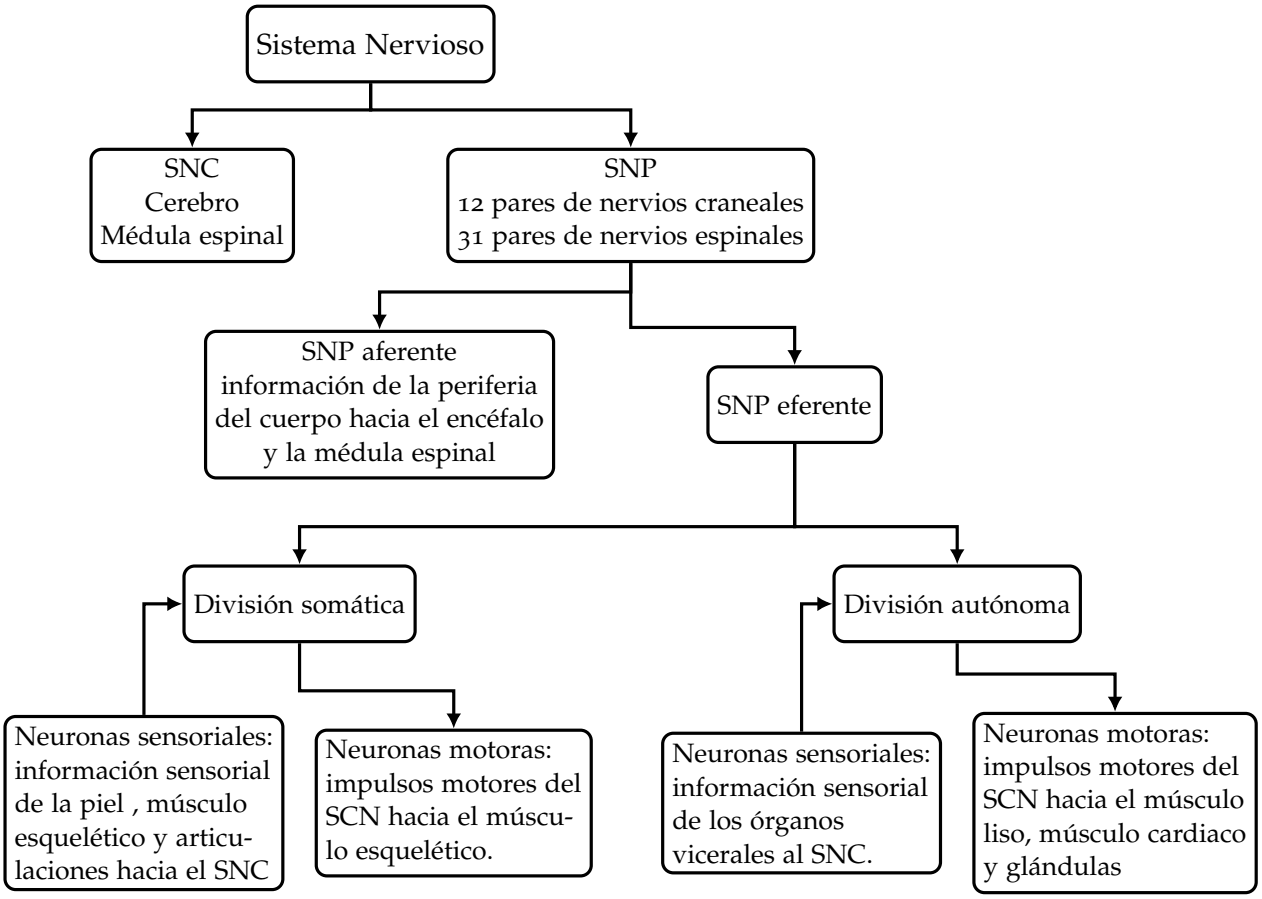


Figura 27: Divisiones principales del sistema nervioso

dritas) y un árbol de salidas (axón y sus terminales).

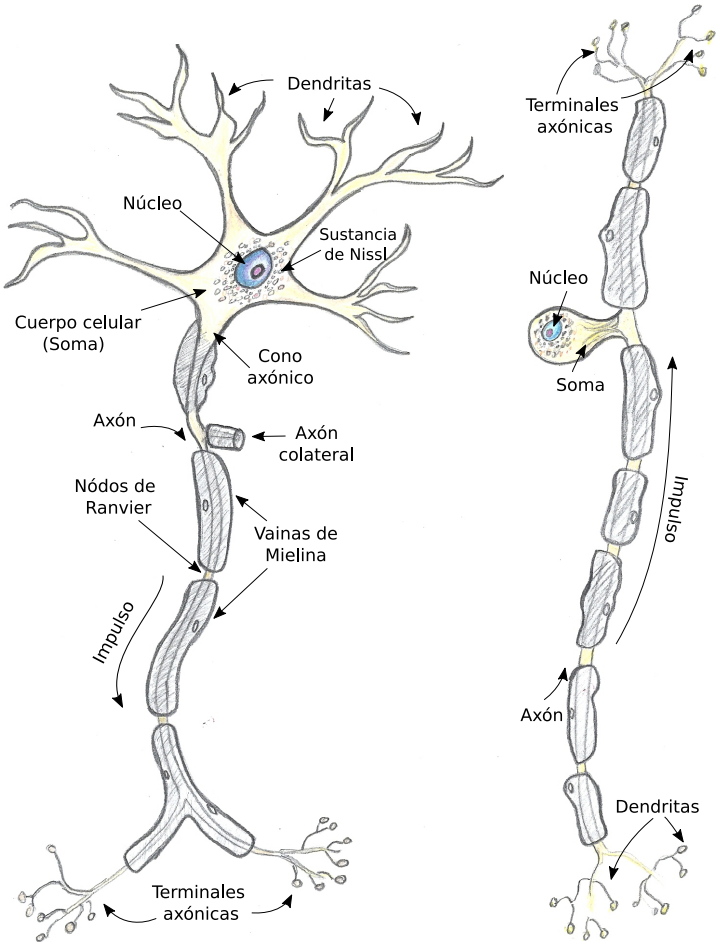
Neuronas unipolares: Solo presentan un proceso derivado del cuerpo celular, una rama central funciona como el axón y una rama periférica como dendrita.

Neuronas bipolares: Solo presentan una dendrita y un axón, y son receptoras en órganos sensoriales especiales. Solo se encuentran en tres áreas del cuerpo humano: la retina del ojo, el oído interno y el área olfatoria de la nariz.

Neuronas multipolares: Es el tipo de neuronas más numeroso del SNC, del cuerpo celular surgen múltiples dendritas que se extienden en todas direcciones, formando así, ramificaciones de aspecto enmarañado.

Las células nerviosas tienen la capacidad de *percibir* los cambios en el medio ambiente a partir de receptores (terminales nerviosas periféricas). Los receptores transforman la energía de los estímulos en impulsos nerviosos, la primera célula en recibirlo es denominada célula neurina aferente o sensorial (es de tipo unipolar), este estimulo se dirige hacia el SNC y en su camino pasa a través de varias células asociativas o internunciales (de tipo multipolar) pertenecientes al cerebro o a la médula.

A partir de la asociación de las neuronas internunciales, la respuesta es transmitida como impulso a la célula nerviosa final o neurona motora (eferente), la cual es de



(a) Multipolar (b) Unipolar

Figura 28: Dos tipos de estructuras neuronales

tipo multipolar y lleva a cabo una reacción ante el estímulo original. Generalmente esta reacción es de tipo muscular, pero también puede ser glandular.

4.3.2 *Conexiones entre las neuronas*

La neurona, desde el punto de vista simplista, no es más que un interruptor con información de entrada y salida. El interruptor solo se activará si el estimulo tiene un valor suficiente, es decir, si las señales conectadas a la entrada *refuerzan* la información provocando que el estimulo alcance un **valor umbral**. Si esto ocurre, el impulso viajará a través del axón y la salida de información se presentará como un impulso para otras neuronas o algún músculo.

Las señales que viajan a través de las neuronas son transferidas mediante conexiones especiales llamadas **sinapsis**. La sinapsis, es en realidad un pequeño espacio, es el área donde las ramas terminales de un axón se anclan cerca (pero sin tocar) las terminaciones de las dendritas de otra neurona. La sinapsis consta de tres elementos fundamentales:

1. Una terminación presináptica que contiene neurotransmisores, mitocondrias y otros orgánulos celulares

2. Una terminal postsináptica que contiene receptores para neurotransmisores
3. Una hendidura sináptica o espacio entre las terminaciones presináptica y postsináptica

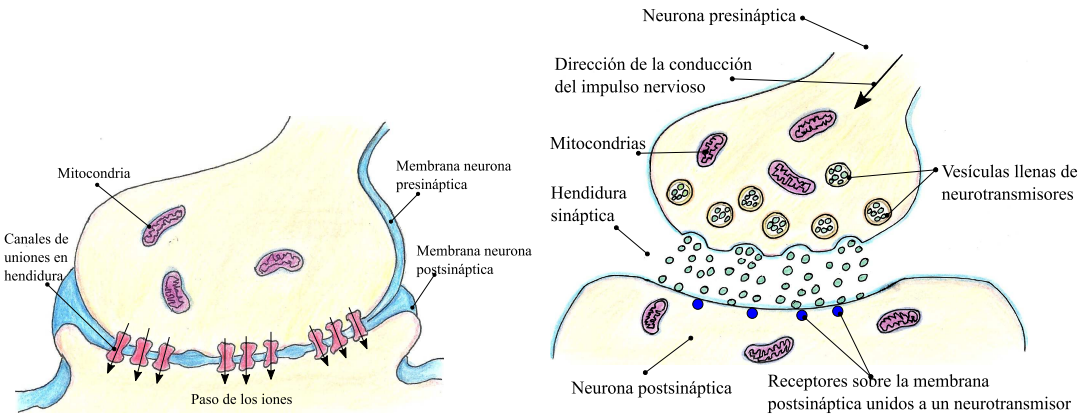
Las conexiones de sinapsis son uniones de una sola vía, que aseguran que la transmisión de información sea en un solo sentido. También, se pueden encontrar otras áreas de sinapsis en el cuerpo, tal como entre las terminales axónicas y los músculos, o entre las terminales axónicas y las glándulas. Por lo tanto, los tipos de sinapsis pueden ser agrupados en dos tipos principales:

SINAPSIS ELÉCTRICA: Es la variante más simple. Una señal eléctrica proveniente, por ejemplo, del lado presináptico se transfiere directamente al núcleo postsináptico de la célula mediante una conexión física y directa entre el transmisor y el receptor de la señal, la cual es fuerte y no ajustable. En este tipo de sinapsis las membranas de las dos células se tocan y comparten proteínas por lo que la señal es muy rápida (ver figura 29-a). Sin embargo, este tipo no es abundante y solo se encuentran en órganos como el corazón y el ojo.

SINAPSIS QUÍMICA: Es la más compleja y más común, aquí no hay un acoplamiento directo como en el caso eléctrico, sino que existe una *hendidura sináptica* que separa eléctricamente el lado presináptico del postsináptico. La transmisión de información se lleva a cabo por secreciones, en muy bajas concentraciones, de químicos conocidos como neurotransmisores que se mueven a través de la hendidura (ver figura 29-b). En el sistema nervioso, existen muchos tipos de neurotransmisores que son clasificados generalmente en dos categorías principales relacionadas con su actividad global: excitadores e inhibidores. Los primeros ejercer efectos excitatorios en la neurona, con ello, el aumento de la probabilidad de que en la neurona se disparare un potencial de acción; algunos de ellos son el glutamato, la epinefrina y la norepinefrina. Por otro lado, los neurotransmisores inhibitorios ejercen efectos inhibidores sobre la neurona y con ello la disminución de la probabilidad de que en la neurona se disparare un potencial de acción; entre ellos se incluyen el GABA, glicina, y la serotonina.

Los neurotransmisores se degradan muy rápido, por lo que es posible liberar pulsos de información muy precisos. A pesar de su funcionamiento más complejo, la sinapsis química tiene en comparación con la sinapsis eléctrica la ventaja de ser una conexión ajustable.

La manera en como las conexiones sinápticas se relacionan con las tres partes principales de la neurona antes mencionadas es la siguiente: Las dendritas se ramifican como arboles desde el cuerpo de la neurona y reciben señales eléctricas de muchas fuentes diferentes, las cuales son después transferidas al cuerpo de la célula (soma). Este último, recibe una gran cantidad de señales de activación e inhibición y las acumula en su interior. Tan pronto como el valor de las señales acumuladas excede un cierto valor (valor umbral) el núcleo de la neurona activa un impulso eléctrico que es transmitido a través del axón. El axón es largo, una extensión esbelta del soma, en un caso extremo, un axón puede extenderse hasta un metro (e.g. dentro de la médula espinal); donde el axón está eléctricamente aislado con el fin lograr una



(a) Sinápsis eléctrica (b) Sinápsis química

Figura 29: Tipos de conexiones neuronales

mejor conducción de la señal eléctrica, así mismo, este puede trasferir información a otro tipo de células con el fin de controlarlas. La salida que proporciona la neurona es de tipo escalar, es decir, la neurona solo aporta un componente de entrada, que a su vez, se une con salidas escalares de otras neuronas para formar la entrada vectorial de otra neurona nueva. Esto significa, que en alguna parte del cuerpo de la neurona los diversos componentes de entrada deben incorporarse para formar solo uno.

4.3.3 *Generación de impulsos*

Unos de los aspectos fundamentales de las neuronas, es el hecho de que el interior de su membrana celular presenta una carga eléctrica diferente al de su exterior, i.e. una diferencia de potencial; esta diferencia de cargas eléctricas es uno de los conceptos más importantes para entender el funcionamiento al interior de una neurona y recibe el nombre de **potencial de membrana.** Una neurona en reposo (que no transmite una señal) posee un cierto valor de potencial natural, conocido comúnmente como: *valor de potencial de membrana en reposo*, o simplemente *potencial de reposo*.

El potencial de membrana se genera debido a concentraciones iónicas características al interior y exterior de la membrana celular. Existen iones sodio (Na^+) con carga positiva, que se encuentra en mayor concentración al exterior de la célula, en cambio, hay una mayor concentración de iones de potasio (K^+), con carga positiva, e iones cloruro (Cl^-), con carga negativa, (así como otras moléculas orgánicas) al interior de la célula. Esta diferencia en la concentración de iones provoca que en ambos lados de la membrana existan gradientes de concentración de los tipos más abundantes de iones (ver figura 30).

Debido a su carga, los iones no pueden pasar libremente de un lado a otro de la membrana celular[3] ya que esta posee regiones de lípidos hidrofóbicos (repelidos

3 Lo que ocurriría naturalmente mediante el proceso de difusión y haría desaparecer los gradientes de concentración

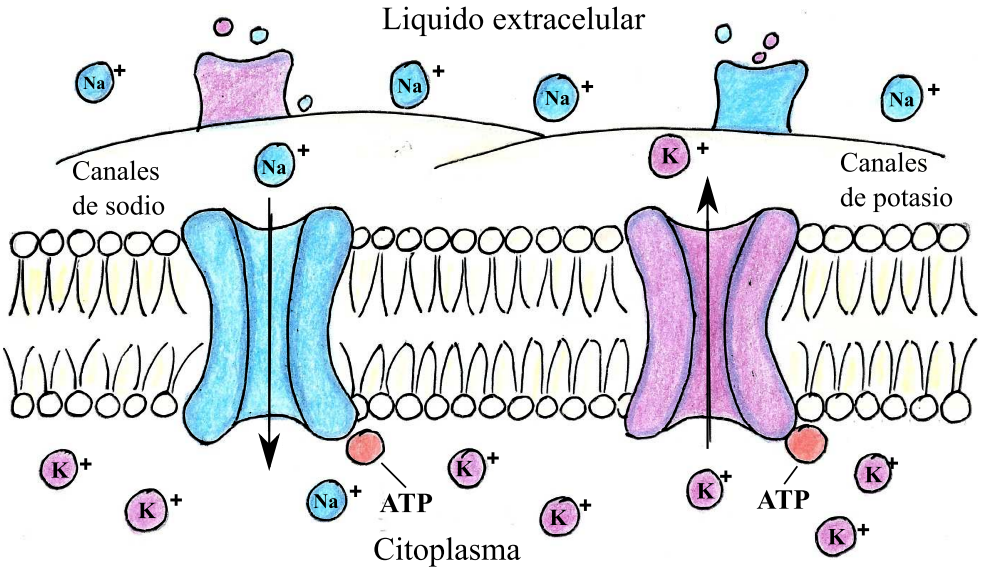


Figura 30: Concentración de iones y canales de filtración

por el agua). En cambio, tienen que utilizar canales de proteína especializados que cruzan la membrana y que proporcionan un túnel hidrofílico (con afinidad por el agua). Algunos canales, llamados canales de filtración, están abiertos en neuronas en reposo, otros solo se abren cuando una señal atraviesa la neurona.

Algunos de estos canales iónicos antes mencionados, son altamente selectivos para un tipo de ion, y otros permiten el paso de varios tipos de iones. Los canales iónicos que permiten principalmente el paso de K^+ se denominan canales de potasio y los que permiten principalmente el paso de Na^+ se denominan canales de sodio.

En las neuronas, el potencial de reposo de membrana depende principalmente del movimiento de K^+ a través de sus canales de filtración. Si se abren canales de potasio en la membrana, el K^+ comenzará a fluir hacia el exterior de la célula; cada vez que un ion de K^+ sale de la célula, el interior de la célula pierde una carga positiva. Por lo tanto, en la parte exterior de la membrana celular se acumula un ligero exceso de carga positiva (+) y en el interior se acumula un ligero exceso de carga negativa (−). Es decir, el interior de la célula se vuelve negativo respecto al exterior y se establece una diferencia de potencial eléctrico en la membrana.

Entre los iones (como en imanes), cargas similares se repelen y cargas diferentes se atraen. Por lo tanto, al establecerse la diferencia de potencial eléctrico en la membrana, se dificulta que los iones de K^+ restantes puedan seguir saliendo de la célula. Los iones K^+ de carga positiva, serán atraídos por las cargas negativas en el interior de la membrana celular y repelidas por las cargas positivas en el exterior, oponiéndose a su movimiento en dirección del gradiente de concentración. Las fuerzas eléctricas y difusivas que rigen el movimiento de K^+ a través de la membrana forman en conjunto su gradiente electroquímico. Finalmente, la diferencia de potencial eléctrico en la membrana celular se acumula hasta un nivel suficientemente alto para que la fuerza eléctrica que impulsa K^+ de regreso a la célula sea igual a la fuerza química que impulsa la salida de K^+.

Cuando la diferencia de potencial en la membrana de la célula llega a este punto, no hay movimiento neto de K^+ en ninguna dirección y el sistema se considera en equilibrio.

En una célula donde solo hay una especie iónica permeante (solo un tipo de iones puede atravesar la membrana) el potencial de reposo será igual al potencial de equilibrio de ese ion (en este caso el K^+). Sin embargo, resulta que la mayoría de las neuronas en reposo son permeables a Na^+ y Cl^-, así como a K^+. Por lo tanto, El Na^+ intentará arrastrar el potencial de membrana hacia su potencial de equilibrio (con valor positivo) y El K^+ intentará arrastrar el potencial de membrana hacia su potencial de equilibrio (con valor negativo). El potencial de membrana real, estará entre el potencial de equilibrio del Na^+ y el potencial de equilibrio del K^+, sin embargo, será más cercano al potencial de equilibrio del ion con mayor permeabilidad, que es comúnmente el K^+. Un valor promedio aceptado para el potencial de membrana en reposo es $-70\ mV$ (con respecto al exterior de la célula)[74].

En este valor de potencial en reposo, ni el Na^+ ni el K^+ están en el potencial de equilibrio que tendrían si fueran el ion único. El potencial de membrana es menos negativo que el potencial de equilibrio del K^+ y al mismo tiempo es menos positivo que el potencial de equilibrio del Na^+. Por lo tanto, aún habrá difusión constante de K^+ hacia el exterior de la célula y de Na^+ hacia el interior. No obstante, la neurona tiene un mecanismo para mantener activamente el potencial de membrana; lo hace mediante una proteína llamada Na^+- K^+ **ATPasa**, que suele conocerse como bomba de sodio-potasio[32] (fig. 31). La bomba Na^+- K^+ es una proteína que atraviesa la membrana celular y que transporta activamente los iones Na^+ y K^+ en contra de sus gradientes electroquímicos. La energía para este movimiento proviene de la hidrólisis de ATP (la división del ATP en ADP y fosfato inorgánico). Por cada molécula de ATP que se rompe 3 iones de Na^+ se mueven del interior hacia el exterior de la célula y 2 iones de K^+ se trasladan del exterior al interior.

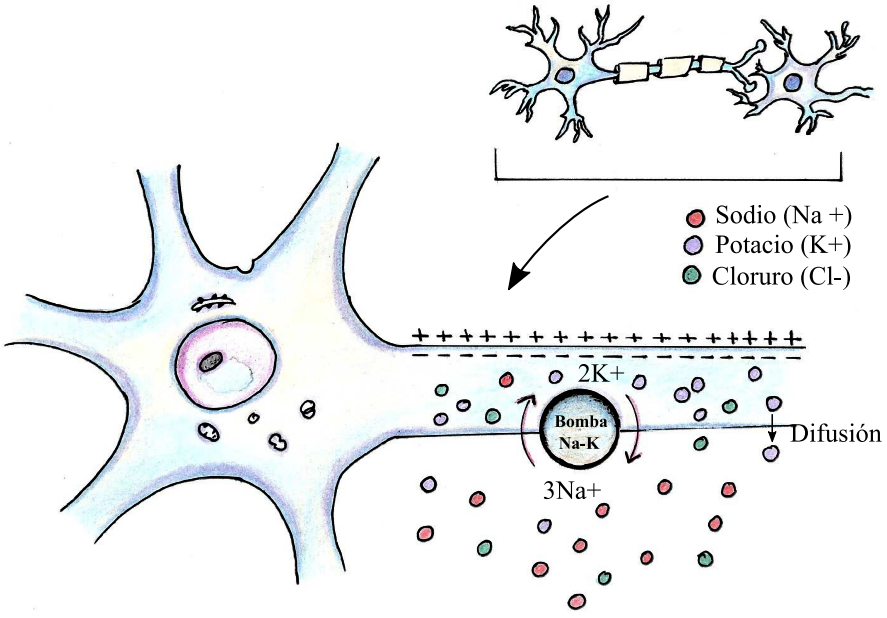


Figura 31: Potencial de membrana y bomba de sodio-potasio

4.3.4 *Activación y transmisión de señales*

Tal como se vió en la §4.3.2, los neurotransmisores cruzan la hendidura sináptica y se unen a receptores especificos en la membrana postsinpática. La acción química en estos receptores produce cambios en la permeabilidad de la membrana postsináptica. Una afluencia de neurotransmisores exitatorios en la célula tenderá a generar una fluencia de iones positivos y por consiguiente despolarizar el potencial de reposo. No obstante, los neurotransmisores inhibidores tenderán a generar la fluencia de iones negativos, produciendo así un efecto de hiperpolarización, ambos efectos son efectos locales que se extienden a corta distancia en el cuerpo celular (soma) y se **suman** en el cono axónico. Si la suma es mayor que un **valor umbral**, se genera un potencial de acción. El valor umbral puede ser diferente en cada tipo de células especializadas, no obstante, tiene un valor promedio aceptado de -55 mV.

Cuando lo anterior ocurre, la neurona comienza a transmitir la señal como un impulso nervioso y la permeabilidad de la membrana celular a lo largo del axón cambia (se abren canales de filtración). El Na^+ entra a la célula cambiando la carga interna de la membrana de negativa ($-$) a positiva ($+$), esta reversión de la carga eléctrica se denomina **despolarización** y crea el **potencial de acción** de la célula. El potencial de acción se mueve en una sola dirección a lo largo del axón y va recorriendo toda la fibra nerviosa (fig. 32).

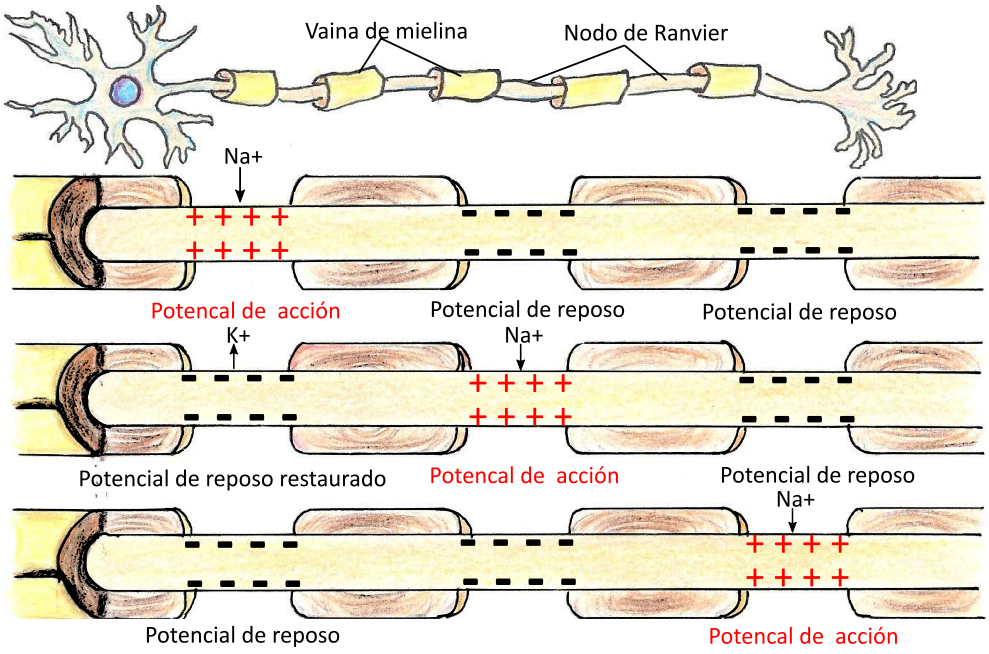


Figura 32: Potencial de acción

Una vez iniciado el potencial de acción, nuevamente la bomba de sodio-potasio comienza a funcionar, bombeando hacia afuera los iones de sodio que entraron en la célula y regresando los iones de potasio que se habían salido; de esta forma se restauran las cargas originales, lo que se conoce como **repolarización** de la membrana, y el interior de la célula vuelve a ser negativo. Este proceso continua a lo largo de la fibra nerviosa, llevando el impuso como una corriente eléctrica a lo largo de la fibra; por lo que se suele decir que el impulso nervioso es una onda de

despolarización autopropagada.

En cualquier fibra nerviosa, el impulso nunca variara en intensidad, esto se conoce como **ley del todo o nada**, que establece que si una fibra nerviosa transporta un impulso, siempre lo llevará hasta su máxima intensidad[65]. En la figura 33 se muestra una gráfica del comportamiento del potencial de membrana cuando la neurona transmite una señal.

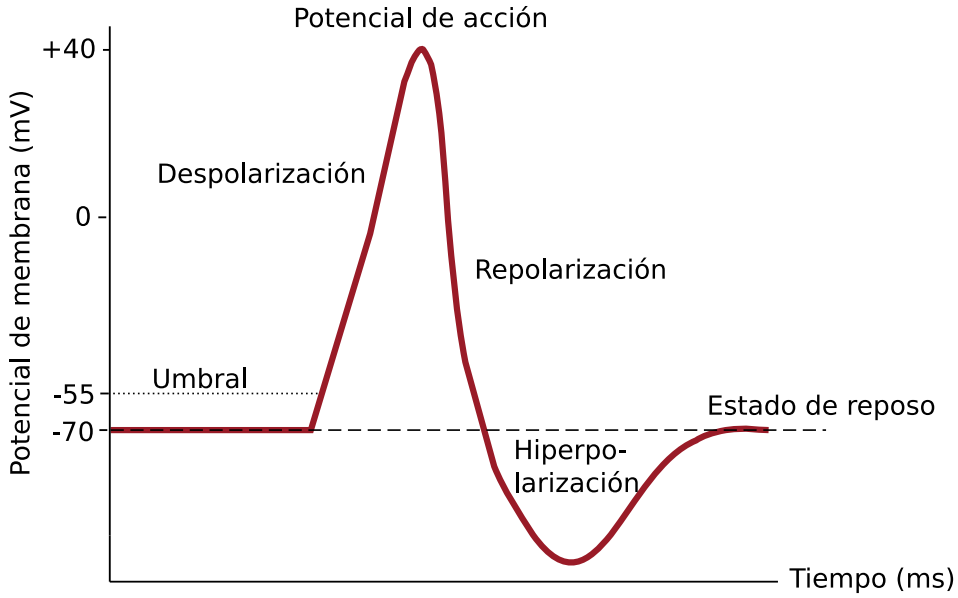


Figura 33: Curva de potencial de membrana[74][36]

.

4.3.5 *Procesamiento de la información*

Gracias a las investigaciones es bien sabido que la naturaleza del procesamiento del cerebro es principalmente de tipo paralelo y distribuido, la información se almacena mediante las conexiones creadas entre las neuronas y las señales se distribuyen a través de la red permitiendo que se procesen en una gran cantidad de neuronas de forma paralela. Así mismo, el cerebro se adapta desde su nacimiento hasta su muerte aprendiendo durante toda su vida mediante modelos que provienen del mundo exterior.

No existe razón para pensar que toda la información recibida por las células sensoriales es transmitida al cerebro para ser procesada ahí, ni que el cerebro es el único que debe asegurarse de que se emitan las salidas correctas en forma de pulsos motores. El procesamiento de la información es completamente descentralizado en el nivel más bajo, por ejemplo, en las células sensoriales la información no se transfiere directamente, sino que existen mecanismos de preprocesamiento que previenen la transmisión continua del estímulo al SNC y aí evitar la *adaptación sensorial*.[4]

4 Debido a la estimulación continua de muchas células receptoras el sistema se vuelve automáticamente insensible a dicho estimulo.

Otro ejemplo son los reflejos que ocurren como una acción involuntaria ante un estímulo externo, como pincharse el dedo con una espina, los reflejos nos permiten responder mucho más rápido a un estimulo sin la necesidad de *pensar* de manera cociente el que hacer ante cada situación.

El arco reflejo es la vía por la que se da un reflejo (Figura 34) es la unidad básica del sistema nervioso y es la vía más simple, pequeña y capaz para recibir un estímulo, entrar al SNC para su interpretación inmediata (por lo general la médula espinal) y producir una respuesta. El arco reflejo tiene cinco componentes:

1. Un receptor sensorial
2. Una neurona sensorial o aferente
3. Neuronas asociativas o internunciales en la médula espinal
4. Una neurona motora o eferente
5. Un órgano efector, como un musculo

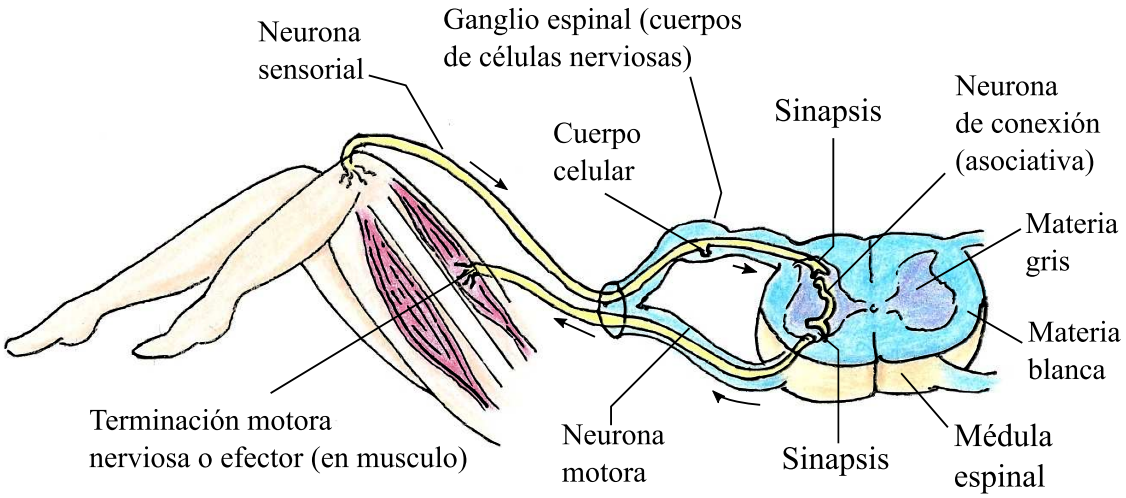


Figura 34: Arco reflejo

Así mismo, en la figura 35 se muestra un esquema del sistema nervioso donde se puede apreciar que el procesamiento de la información es descentralizado y existen diferentes redes neuronales dedicadas al procesamiento y transmisión de información de sistemas independientes.

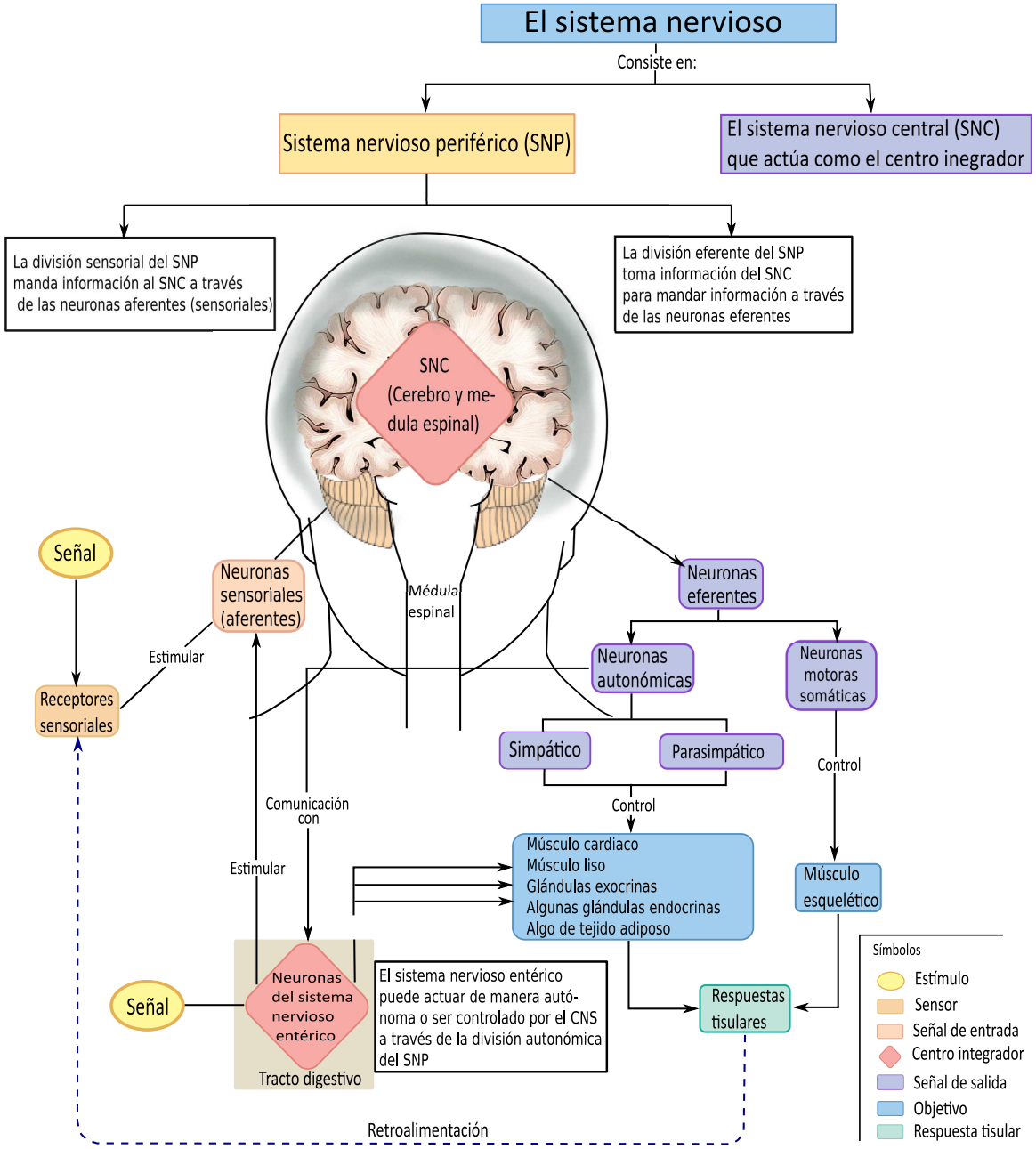


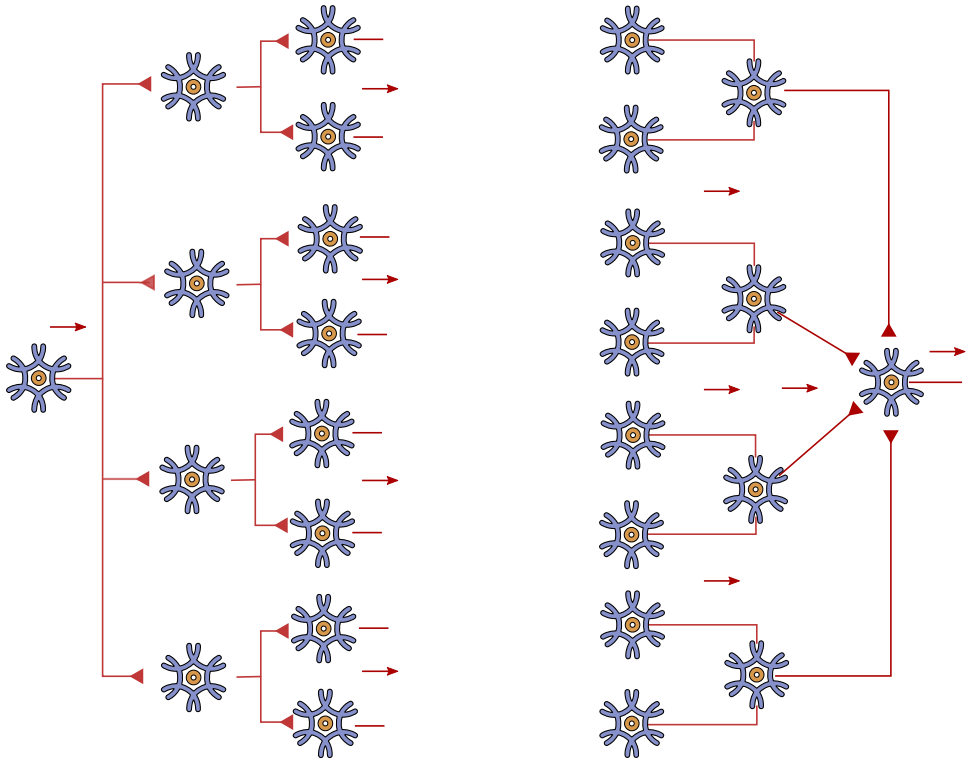
Figura 35: Organización del sistema nervioso[74]

4.3.5.1 *Convergencia y divergencia*

La divergencia y la convergencia de las conexiones neuronales son un principio básico de la organización del cerebro. Por ejemplo, la divergencia permite que la información recogida por un único receptor sensorial se distribuya en muchas áreas del cerebro. La convergencia, por su parte, permite que la suma de información de muchas áreas la reciba un único receptor, por ejemplo las neuronas que se encargan de contraer la musculatura, que reciben la suma de información de muchas neuronas.

Se Habla de divergencia sináptica cuando la información de un axón se transmite a muchas neuronas postsinápticas. De este modo, se amplifica la información (ver figura 36-a). Por otro lado, se habla de convergencia sináptica cuando varios nodos

terminales hacen sinapsis sobre una misma neurona y estas informaciones convergentes se integran en una sola respuesta postsináptica (figura 36-b).



(a) Divergencia (b) Convergencia

Figura 36: Esquema de los procesos de divergencia y convergencia sináptica

4.3.6 *Cantidad de elementos neuronales*

El procesamiento en paralelo de millones de células neuronales es la razón de porque este tipo de sistemas son tan poderosos; la división masiva de trabajo le permiten tener cualidades de eficiencia y tolerancia a errores. En la naturaleza es posible encontrar una gran variedad de redes neuronales en los organismos vivos, si bien el funcionamiento de la red no difiere mucho de especie a especie si lo hace el número de elementos que puede contener, esto está directamente relacionado a las capacidades cognitivas, velocidad de respuesta y tamaño del cuerpo del ser vivo en particular. Así mismo, las capacidades cognitivas de un individuo en particular pueden potenciarse al estar inmerso en un grupo colaborativo, tal es el caso de las hormigas y abejas que al comportarse como un todo pueden resolver problemas, que superan por mucho su capacidad individual. A continuación, se presenta en la tabla 10 algunos ejemplos de la capacidad neuronal cortical[5] de diferentes seres vivos.

5 Referente a la corteza del cerebro, sin contar la médula espinal y el demás sistema nervioso.

Tabla 10: Cantidad de neuronas corticales en organismos vivos[36]

Número de Neuronas	Ser vivo	Número de Neuronas	Ser vivo
302	Requeridas para el sistema nervioso de un **gusano nematodo**, un organismo modelo muy popular en biología	$5 \cdot 10^7$	Pertenecen a un **murciélago**, ellos navegan en total obscuridad mediante ecolocación y atrapan a su presa en pleno vuelo
10^4	Necesarias para una **hormiga**, que puede participar en comportamientos sociales complejos y formar colonias con millones de individuos.	$1.6 \cdot 10^8$	Necesarias para el cerebro de un **perro**, fiel e inseparable amigo del hombre, pueden aprender, pensar y resolver problemas. También, poseen inteligencia de trabajo y obediencia.
10^5	Conforman el sistema de una **mosca**,la cual puede evadir un objeto en tiempo real en un espacio tridimensional, puede aterrizar boca abajo en el techo, tiene un sistema sensorial considerable debido a ojos compuestos y mucho más	$3 \cdot 10^8$	Es el número encontrado en un **gato**, ellos presentan un comportamiento paciente y elegante, Además poseen reflejos especiales como el reflejo de enderezamiento que les permite caer sin lastimarse. Un **pulpo** podría colocarse en la misma magnitud neuronal.
$0.8 \cdot 10^6$	Suficiente para una **abeja melífera**, que construye colonias y tienen capacidades increíbles en el campo del reconocimiento aéreo y la navegación.	$3.4 \cdot 10^9$	Es el número en un **Chimpancé**, uno de los animales más parecidos al humano, son capaces de tener complejos sistemas sociales dentro de su comunidad
$4 \cdot 10^6$	Necesarias para el sistema de un **ratón**, el cual puede navegar por laberintos, localizar alimento y huir de los depredadores, con el se da inicio al mundo de los vertebrados.	10^{10}	Suficientes para conformar a un ser **humano**, el cual tiene capacidades cognitivas considerables, es capaz de hablar, abstraer, recordar y utilizar herramientas, así como el conocimiento de otros humanos para desarrollar tecnologías avanzadas y múltiples estructuras sociales.
$1.5 \cdot 10^7$	Son suficientes para una **rata**, que a menudo se utilizan para participar en una variedad de pruebas de inteligencia. Las ratas tienen un sentido del olfato y orientación extraordinarios, y también muestran un comportamiento social. El cerebro de una **rana** puede ponerse en la misma dimensión de neuronas.	$2 \cdot 10^{11}$	Sistemas con más neuronas que el sistema nervioso humano, tales como **elefantes** y **ballenas**, sus grandes cuerpos requieren una red mucho más numerosa para realizar su control.

4.4 Redes neuronales artificiales

Las funciones cerebrales de los seres vivos, tales como el procesamiento de la información sensorial y la cognición son el resultado de cálculos emergentes realizados por la masiva red neuronal existente. Las redes neuronales artificiales, son modelos computacionales inspirados por el principio de cálculo de las redes neuronales biológicas, las redes neuronales [6] poseen muchas características atractivas que, en última instancia, pueden superar algunas de las limitaciones en los sistemas computacionales clásicos.

Una neurona biológica es capaz de producir un tren rápido de picos eléctricos.Su compleja dinámica interna ha sido descrita por el modelo de Hodgkin-Huxley que tiene en cuenta la morfología tridimensional exacta de la célula. Sin embargo, simular un modelo tan preciso es extremadamente exigente en lo que respecta a potencia computacional, y aunque es de gran interés para la investigación del cerebro es poco práctico para las aplicaciones del mundo real. Por esta razón, para simular el comportamiento de las neuronas se ha mantenido el modelo del comportamiento de los picos eléctricos, pero se simplifica en gran medida la dinámica interna. Dichas neuronas simplificadas son denominadas neuronas analógicas, esta simplificación elimina la dinámica temporal compleja de los procesos biológicos y hace posible cálculos en tiempo discreto, y esto a su vez permite simular grandes cantidades de neuronas con un poder computacional relativamente bajo.

Tal como se vio en la §4.3.1, las neuronas biológicas están conectadas entre sí de forma ponderada, cuando son estimuladas transmiten eléctricamente su señal a través del axón. Desde el axón, las señales no se transfieren directamente a las neuronas siguientes, primero se tiene que cruzar la hendidura sináptica donde la señal cambia a una señal química variable, en la neurona receptora, las diversas entradas adquiridas son procesadas y posteriormente se acumulan en un solo pulso, dependiendo del estimulo de entrada, la neurona emite un pulso o no. Por lo tanto, la salida es de naturaleza no lineal y no es proporcional a la entrada acumulada; en la tabla 11 y en la §4.4.1 se presenta un breve resumen de los elementos esenciales de las redes neuronales biológicas, estos serán incluidos a continuación en el modelo equivalente para las neuronas artificiales:

Tabla 11: Equivalencia entre elementos de la neurona biológica y artificial

Elemento	Neurona biológica	Neurona artificial
Entradas	Señal de dendritas	Vector de datos
Valores ajustables de entrada	Tipos de neurotransmisores	Pesos sinápticos
Acumulación de entradas	Acumulación en el cono axónico	Suma ponderada
Modalidad de activación	Valor umbral	función de activación
Naturaleza no lineales	La salida no es proporcional a la entrada	funciones no lineales
Salida	Un solo impulso nervioso	Valor escalar

6 A menos que se indique lo contrario, se utilizará el término red neuronal o ANN para hacer referencia a las redes neuronales artificiales.

4.4.1 *Componentes de las redes neuronales*

ENTRADAS: En biología, los valores de entrada ocurren en las dendritas donde cada neurona presináptica aporta señal de entrada, en las neuronas artificiales estas entradas se visualizan como una cantidad vectorial ($\vec{x}$) que contiene los datos a procesar; en la naturaleza, una neurona recibe de 10^3 a 10^4 impulsos en promedio provenientes de otras neuronas.

VALORES AJUSTABLES: Los neurotransmisores contienen información de la intensidad y el tipo de señal (excitación o inhibición), así mismo, las neuronas artificiales utilizan valores llamados **pesos sinápticos** que ponderan las entradas y que son ajustables ($\vec{\omega}$). Por lo tanto, la fuerza (o peso) de una conexión entre dos neuronas i y j es referido como ω_{ij}. Esto agrega una gran dinámica a la red, ya que una gran parte del conocimiento se guarda en estos pesos así como sucede mediante procesos químicos en la hendidura sináptica.

ACUMULACIÓN DE ENTRADAS: En las redes artificiales, las entradas también se preprocesan, estas se acumulan, lo que se refleja en el aspecto técnico como una suma ponderada (S). Esto significa que después de la acumulación se obtiene un solo valor, un escalar, en lugar de un vector.

$$S = \sum_{i=1}^{N} \omega_i \cdot x_i \tag{4.1}$$

MODALIDAD DE ACTIVACIÓN: Una neurona comenzará a transmitir una señal cundo su valor umbral sea alcanzado, esto es representado mediante una función de activación $y = f(S)$ que al igual que las neuronas biológicas es de naturaleza no lineal por lo que la salida no es proporcional a la entrada. La función de activación, define el comportamiento individual de las neuronas, es decir, cómo responden a las señales de entrad; la función elegida por el investigador esta estrechamente relacionada con la aplicación.

Una neurona con una función de activación umbral se denomina perceptrón discreto, y una neurona con una función de activación continua (generalmente una función sigmoidea) se denomina perceptrón continuo. La función sigmoidea, es la función de activación más prevalente y biológicamente plausible, es una de las opciones más populares, junto con la llamada función rectificadora lineal. También, se utilizan otras funciones como la tangente hiperbólica o el seno. A continuación, se muestran algunas de las funciones de activación comúnmente empleadas para las neuronas:

■ Función de activación *umbral* (función de paso unitario o limitador duro):

$$f(S) = \begin{cases} 1.0, & \text{cuando } S > 0 \\ 0.0, & \text{otra} \end{cases}$$

■ Función de activación de *rampa*:

$$f(S) = max\{0.0, min\{1.0, u + 0.5\}\}$$

■ Función de activación *sigmoide*

función sigmoide unipolar:

$$f(S) = \frac{a}{1 + \exp(-bu)}$$

y sigmoide bipolar:

$$f(S) = a\left(\frac{1 - \exp(-bu)}{1 + \exp(-bu)}\right)$$

Donde a y b representan constantes reales, la ganancia y la pendiente de la función de transferencia respectivamente.

SALIDA ESCALAR: Entonces el mapeo no lineal f define la salida escalar y:

$$y = f\left(\sum_{k=1}^{N} \omega_i \cdot x_i\right) \tag{4.2}$$

Que tiene el equivalente biológico de un solo pulso nervioso que se transmite a las siguientes neuronas de la red. En la figura 37 se muestra el esquema de una neurona artificial donde se pueden apreciar todos los componentes anteriormente descritos.

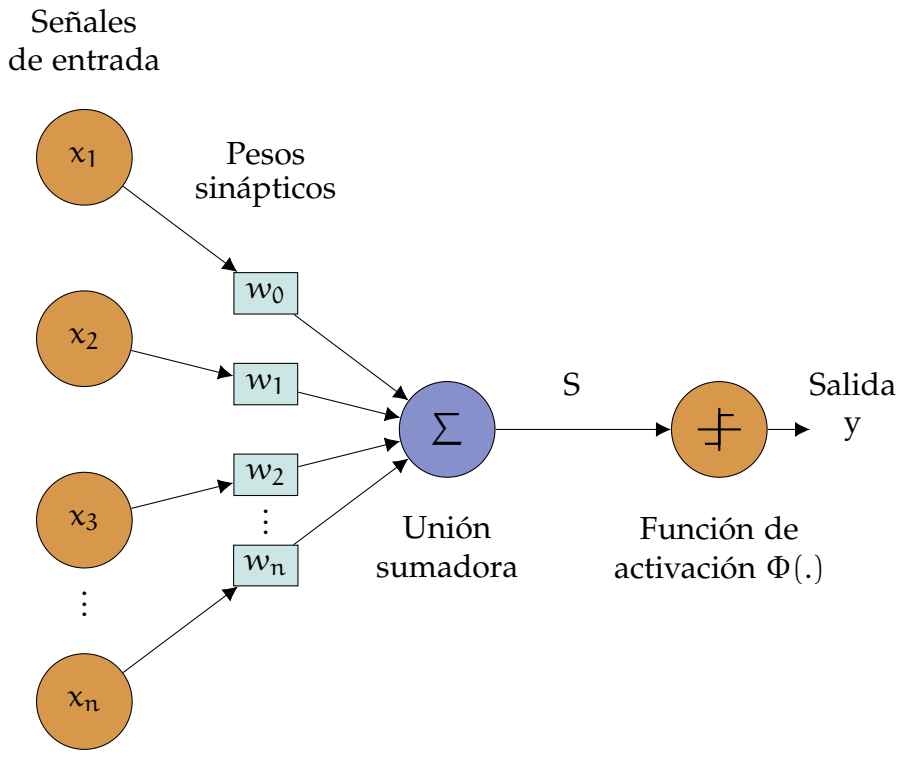


Figura 37: Componentes básicos de una neurona artificial[36]

4.4.2 *Procesamiento de datos en la neurona*

Los datos son transferidos entre las neuronas a través de conexiones ponderadas ya sea bien excitadoras o inhibidoras, el procesamiento de datos dentro de una neurona puede modelarse mediante tres funciones que transforman la información recibida (ver fig. 38)

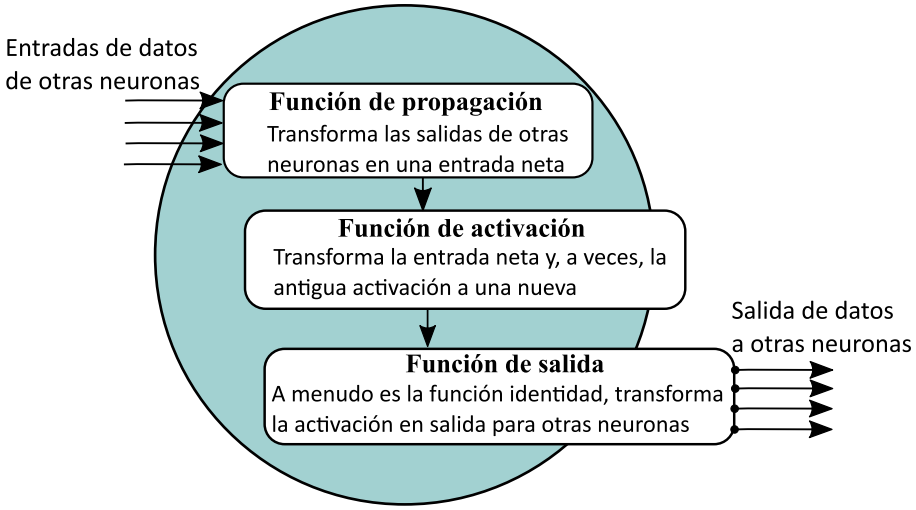


Figura 38: Procesamiento de datos de una neurona[36].

FUNCIÓN DE PROPAGACIÓN: La función de propagación convierte la entrada vectorial a una entrada de tipo escalar, el uso de la suma ponderada es muy popular como función de propagación: la salida de cada neurona i es multiplicada por un peso ω_{ij} y la sumatoria es el resultado (ver la Ec. 4.1).

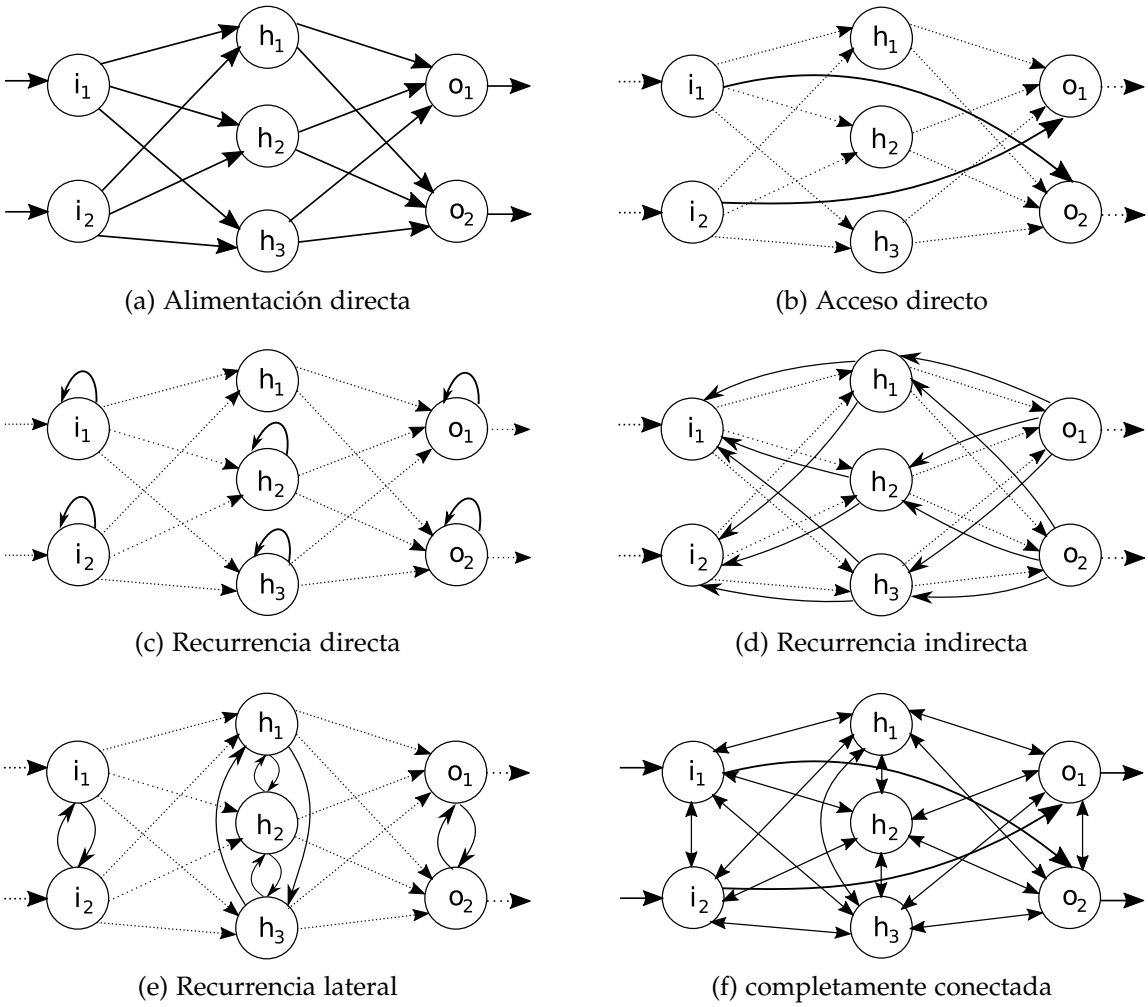
FUNCIÓN DE ACTIVACIÓN: La activación es el estado de cambio de una neurona, la reacción de las neuronas a los valores de entrada dependen de este estado de activación. Desde el punto de vista biológico, el valor umbral representa el valor en el que una neurona comienza a conducir; el valor umbral normalmente es incluido en la definición de función de activación, la cual depende de la entrada de la red y el valor umbral; la función de activación es llamada también función de transferencia.

FUNCIÓN DE SALIDA: Se puede utilizar una función de salida para procesar la información una vez más, la función de salida de una neurona calcula los valores que se transferirán a las otras neuronas conectadas. En general, la función de salida se define globalmente y es la función identidad.

4.4.3 *Topología de la red*

La forma en que se estructuran las neuronas de una red neuronal está íntimamente relacionada con su comportamiento y el algoritmo de aprendizaje utilizado para entrenar la red, en general, se pueden distinguir seis diferentes tipos de topología que se muestran en la fig. 39. Par evitar confusión, se denota con la letra i a las

neuronas de la capa de entrada, con h a las que pertenecen a capas intermedias y con o a las neuronas de la capa de salida.



(a) Alimentación directa

(b) Acceso directo

(c) Recurrencia directa

(d) Recurrencia indirecta

(e) Recurrencia lateral

(f) completamente conectada

Figura 39: Tipos de topología de red[36]

ALIMENTACIÓN DIRECTA: Conocidas en inglés como *Feedforward networks*, las capas neuronales de está red (fig.39-a) están claramente separadas: contiene una capa de entrada, una capa de salida y una o más capas ocultas o de procesamiento que son invisibles desde el exterior, solo existen conexiones hacia las neuronas de la capa sucesora.

CONEXIONES DE ACCESO DIRECTO: Similares a las redes de alimentación directa, pero las conexiones pueden no solo dirigirse hacia la capa sucesora sino también hacia cualquier otra capa predecesora (fig. 39-b).

RECURRENCIA DIRECTA: Algunas redes permiten que las neuronas se conecten entre sí, lo que se denomina recurrencia directa o autorrecurrencia; como resultado las neuronas inhiben o fortalecen su conexión para alcanzar sus valores umbrales de activación (fig. 39-c).

RECURRENCIA INDIRECTA: En este tipo de redes se permiten conexiones hacia la capa de entrada, la red ahora contiene conexiones adicionales entre las neuronas y su capa anterior (fig. 39-d).

RECURRENCIA LATERAL: Una red lateralmente recurrente permite conexiones dentro de una capa, aquí cada neurona a menudo inhibe las otras neuronas de la capa y se fortalece. Como resultado, solo se activa la neurona más fuerte (el ganador se lo lleva todo) (fig. 39-e).

COMPLETAMENTE CONECTADAS: En este caso, se permite que cada neurona se conecte a cualquier otra neurona (fig. 39-f), pero como resultado cada neurona puede convertirse en una neurona de entrada. Por lo tanto, las recurrencias directas normalmente no se pueden aplicar aquí y las capas claramente definidas ya no existen. Además, las conexiones deben ser simétricas.

4.4.3.1 *Orden de activación de las neuronas*

En las redes neuronales también es importante en que orden se activan las neuronas individuales, es decir, en que orden procesan la información de entrada y dan una respuesta; se pueden distinguir en principio dos modelos de activación[36]: síncrono y asíncrono. En el primero, todas las neuronas cambian sus valores de forma sincronizada, es decir, calculan simultáneamente las entradas; la activación síncrona es correspondiente a la activación que ocurre en las redes biológicas, es útil sobretodo si se implementan en hardware y en ciertas computadoras paralelas, este orden de activación es el más genérico y puede usarse con redes de topología arbitraria.

Por otro lado, en la activación asíncrona las neuronas no cambian sus valores simultáneamente, sino en diferentes momentos, para lograr esto existen diferentes ordenes de activación como los siguientes:

ORDEN ALEATORIO: Una neurona es seleccionada aleatoriamente y tanto su activación como su salida son actualizadas, para una red con **n** neuronas, un ciclo corresponde a realizar n veces el paso anterior. Por consiguiente, en un ciclo algunas neuronas serán actualizadas repetidas veces, mientras que otras no tanto.

PERMUTACIÓN ALEATORIA: Cada neurona es seccionada en orden aleatorio pero una sola vez durante un ciclo, este orden de activación se usa raramente porque el orden generalmente es inútil y es muy lento calcular la nueva permutación para cada ciclo.

ORDEN TOPOLÓGICO: Las neuronas se actualizan durante el ciclo y de acuerdo con un orden fijo que está definido por la topología de la red.

4.4.4 *Aprendizaje y entrenamiento*

Las redes neuronales pueden verse como un sistema de procesamiento masivamente paralelo formado por unidades individuales, que tiene una propensión natural

a almacenar el conocimiento experimental, uno de los aspectos más importantes mediante el cual se establece un conocimiento a partir de la información obtenida[7] es realizar un proceso de entrenamiento en el cual se utiliza la información del entorno y se modifica la red sistemáticamente para *grabar* en ella la tarea que debe asimilar.

Teóricamente una red neuronal puede obtener conocimiento mediante los siguientes paradigmas de aprendizaje:

1. Generando nuevas conexiones
2. Borrando conexiones existentes
3. Modificando los pesos sinápticos entre las conexiones
4. Modificando los valores umbrales de las neuronas
5. Variando una o más de las tres funciones neuronales (función de activación, función de propagación o función de salida)
6. Generando nuevas neuronas
7. Borrando neuronas existentes junto con sus conexiones

Sin embargo el punto 3, la modificación del valor de los pesos sinápticos, es el paradigma más utilizado, los otros son utilizados en aplicaciones especificas donde se puedan aprovechar ventajas particulares del paradigma elegido. Por lo tanto, el procedimiento de aprendizaje es siempre un algoritmo que puede ser clasificado en cualquiera de los tipos de aprendizaje automático mencionados en la §4.1. Bajo el paradigma de aprendizaje supervisado por ejemplo, la red se alimenta con numerosas instancias de entrada, y la salida se compara con una salida deseada. También, se pueden usar varios criterios de convergencia para ajustar los pesos y que la señal de salida coincida lo más posible con la salida deseada.

El aprendizaje de un perceptrón continuo se realiza mediante el ajuste del vector de pesos (utilizando un procedimiento de descenso de gradiente), a través de la minimización de alguna función de error, generalmente se toma el error cuadrado entre la salida deseada y la salida real, los pesos del perceptrón se inicializan mediante valores aleatorios, y se considera que se ha logrado la convergencia del algoritmo cuando no se producen más cambios significativos en el vector de pesos.

Una red neuronal obtiene su poder computacional de su organización paralela y distribuida así como de su capacidad de aprender y generalizar, la generalización hace referencia a la capacidad de la red para producir salidas razonables para entradas que no se vieron durante el entrenamiento. Las características de generalización proporcionan a las redes neuronales la posibilidad de resolver problemas complejos que en la actualidad aún son intratables[36].

7 Entiéndase como conocimiento a la mezcla de experiencia, valores, información y saber hacer. El conocimiento se deriva de la información, así como la información se deriva de los datos.

4.4.5 *Perceptrones de Rosenblatt*

El modelo matemático más simple de una neurona es el perceptrón, uno de ellos es el perceptron de Rosenblatt que esta formado por tres capas: una capa sensorial (**S**) o de entrada, una capa asociativa (**A**) o intermedia y una capa de reacción (**R**) o de salida. En el pasado, muchas investigaciones fueron dedicadas a perceptrones que contenian una sola neurona en la capa R. Tales perceptrones pueden reconocer solo dos clases[8]. Si la salida de la neurona R es superior a un umbral predeterminado (T), la información de entrada pertenece a la clase #1, y si es inferior a (T), la información de entrada entonces pertenece a la clase #2. La estructura de este perceptrón se utiliza sobretodo para diferenciar algún objeto de su fondo en una imagen, sus tres capas funcionan de la siguiente manera: Las neuronas de la capa sensorial (capa S) tienen dos estados $\{1, 1\}$, estas se establece en 1 si la información de entrada corresponde al objeto en la imagen y se establecen en 1 si pertenece al fondo.

La capa asociativa (A) contiene neuronas con dos estados de salida $\{0, 1\}$. Las entradas de estas neuronas están conectadas con las salidas de las neuronas de la capa S (sin conexiones modificables). Cada conexión puede tener su peso con valor igual a 1 (conexión positiva) o 1 (conexión negativa), dejando que el umbral de tales neuronas sea igual al número de sus conexiones de entrada. Estas neuronas se activarán solo si todas las conexiones positivas recibidas corresponden al objeto de la imagen y todas las conexiones negativas corresponden al fondo.

La neurona R de salida se conecta con todas las neuronas de la capa A y los pesos de estas conexiones son modificables durante el entrenamiento; la regla de entrenamiento más popular es aumentar el valor de los pesos entre las neuronas activas de la capa A y la neurona R si el objeto pertenece a la clase 1. Si el objeto pertenece a la clase 2, el valor de los pesos correspondientes se disminuyen. Se sabe que estos perceptrones tienen convergencias rápidas y pueden formar superficies discriminatorias no lineales, la complejidad de las superficies discriminantes depende del número de neuronas de la capa A.[44].

EL PERCEPTRÓN MULTICAPA El perceptrón multi-capa (MLP, por sus siglas en inglés) es una red neuronal de alimentación directa que consta de una capa con nodos de entrada (S), seguida de dos o más capas de perceptrones como el de Rosenblatt, y una capa de salida (R). Las capas que se encuentran entre la capa de entrada y la capa de salida se denominan capas ocultas o intermedias (I). Los MLP se han aplicado con éxito a muchos problemas complejos del mundo real, aunque pueden contener muchas capas se ha comprobado a través del tiempo que los MLP con tres capas son más que suficientes para la mayoría de las aplicaciones.

8 El numero de neuronas en la capa R es igual al número de clases que puede reconocer el perceptrón

4.5 Reconocimiento de patrones

En la mayoría de las aplicaciones de reconocimiento con sistemas de visión artificial, la tarea de clasificación de imágenes es una de las más importantes, de manera general el procedimiento se basa en el uso de tres criterios: el color, la suavidad y la compacidad, algunas técnicas se basan en la extracción de características de la imagen, con esta técnica se utilizan descriptores para obtener las propiedades de color, textura y forma, donde la clasificación por segmentación de texturas juega un papel importante en el reconocimiento de objetos (también llamados patrones). En la figura 40 se muestra un diagrama con las partes generales para el reconocimiento de una imagen. El extractor de características, transforma la imagen en un vector de parámetros, cada componente de este vector corresponde a una característica específica utilizada para resolver diferentes problemas de reconocimiento. En el último bloque, se encuentra el clasificador de imagen, el cual consta de un algoritmo que hace uso del vector de parámetros para reconocer y determinar lo que se encuentra en la imagen.

A pesar de su gran importancia, no existe una definición formal de *textura* debido a la infinita diversidad de muestras existentes. Sin embargo, se puede entender como textura a la propiedad de una sección local de la imagen, que se observa en una sección extensa de la imagen[44]. Algunos ejemplos de secciones que presentan textura idéntica en una imagen son: el césped, el asfalto, el follaje de arboles, etc.

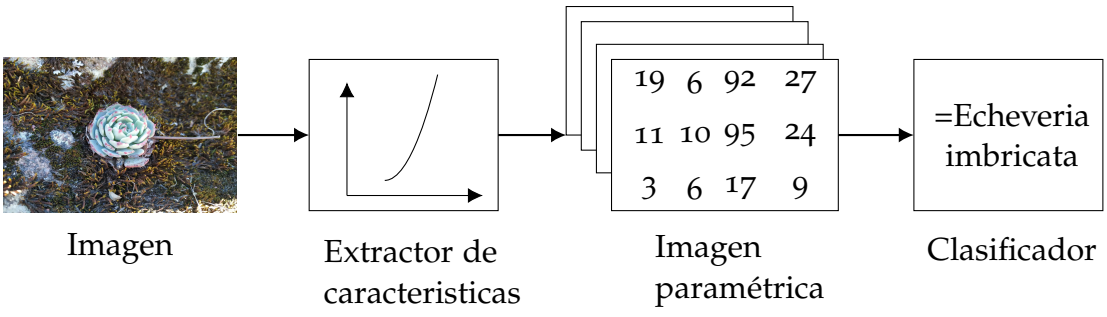


Figura 40: Estructura de un sistema de reconocimiento

Los algoritmos de análisis de textura se utilizan en diferentes áreas, por ejemplo, en la industria electrónica, mecánica y de manufactura, para la inspección de alimentos, para el análisis de imágenes satelitales, etc. Existen diferentes enfoques para resolver el problema de reconocimiento de texturas, algunos de ellos se basan en el análisis de regiones muy pequeñas. Sin embargo, si se utiliza la arquitectura de las redes neuronales para resolver esta tarea, se logra aportar al sistema importantes propiedades de adaptabilidad y robustez frente a la gran variedad de texturas[44].

Desde 1970 el Dr. Ernst Kussul y la Dra. Tetiana Baydik han investigado y desarrollado diferentes clasificadores neuronales, los cuales se han utilizado para el reconocimiento de patrones y en sistemas de control para robots móviles autónomos. Además, estos clasificadores han demostrado buenos resultados en tareas de reconocimiento de textura, en tareas de verificación de identidad basadas en voz, en escritura, reconocimiento de rostros, y en el área de micromecánica. Algunos de

sus clasificadores neuronales más interesantes son el clasificador de umbrales aleatorios (RTC, por sus siglas en inglés) [37, 43], el clasificador de subespacio aleatorio (RSC, por sus siglas en inglés) [40, 42], el clasificador de área de recepción limitada (LIRA, por sus siglas en inglés) [39, 42] y el clasificador neuronal de permutaciones de código (PCNC, por sus siglas en inglés).

4.6 Clasificador neuronal LIRA

Haciendo un análisis de los clasificadores neuronales desarrollados por el Dr. Kussul y la Dra. Baydik mencionados anteriormente, se pueden clasificar dentro de dos tipos de sistemas de reconocimiento de imágenes: el primer tipo, son los que contienen en su estructura un extractor de características y un clasificador (igual a la Fig. 40). el segundo tipo, contiene solo el clasificador: los sistemas de reconocimiento de imágenes como el perceptron de Rosenblatt [66], entre otros, pertenecen también a este último tipo de sistema, estos, se caracterizan en que el extractor de características esta contenido intrínsecamente en su estructura neuronal[7].

El clasificador neuronal **LIRA**, es una red neuronal artificial basada en el Perceptrón de Rosenblatt [66], pertenece al segundo tipo de sistemas de reconocimiento de imágenes; fue desarrollado y propuesto por el Dr. Enrnst Kussul y la Dra. Tetyana Baydyk en [39] y [38].

Existen dos modalidades de este clasificador y se diferencian por las características de la imagen que se presenta a la entrada del clasificador. Si la imagen se presenta en forma binaria (valores blanco y negro), el clasificador recibe el nombre de *LIRA binario*, así mismo, si la imagen se presenta con valores en escala de grises entonces se denomina *LIRA en escala de grises*.

El clasificador neuronal LIRA ha sido probado en tareas de reconocimiento de cifras escritas a mano [39], clasificación de imágenes de micro tornillos [51] y tareas de ensamble de microdispositivos [38] demostrando en ellas buenos resultados. El clasificador LIRA se describe en detalle en [44] y [41]. En este trabajo solo se explicarán las características mas generales.

4.6.1 *Descripción del clasificador LIRA*

EL clasificador LIRA consta de tres capas al igual que el perceptrón de Rosenblatt, sin embargo en [51] se describen algunas modificaciones a su estructura y algoritmos propuestas por el Dr. Kussul. Por ejemplo, para adaptar el clasificador LIRA para el reconocimiento de imágenes en escala de grises, se agrega una capa de neuronas adicional que recibe el nombre de *capa intermedia*. Por lo tanto, podemos hablar que LIRA-escalas de grises consta de cuatro capas (Fig. 41), estas son: capa sensorial (S), capa intermedia (I), capa asociativa (A) y capa de respuesta (R).

CAPA SENSORIAL (S): Corresponde a la *retina*, en términos técnicos corresponde a la imagen de entrada; la imagen de entrada se toma como un ventana con dimensiones en píxeles (HxW), este tamaño se define según la tarea a realizar. Den-

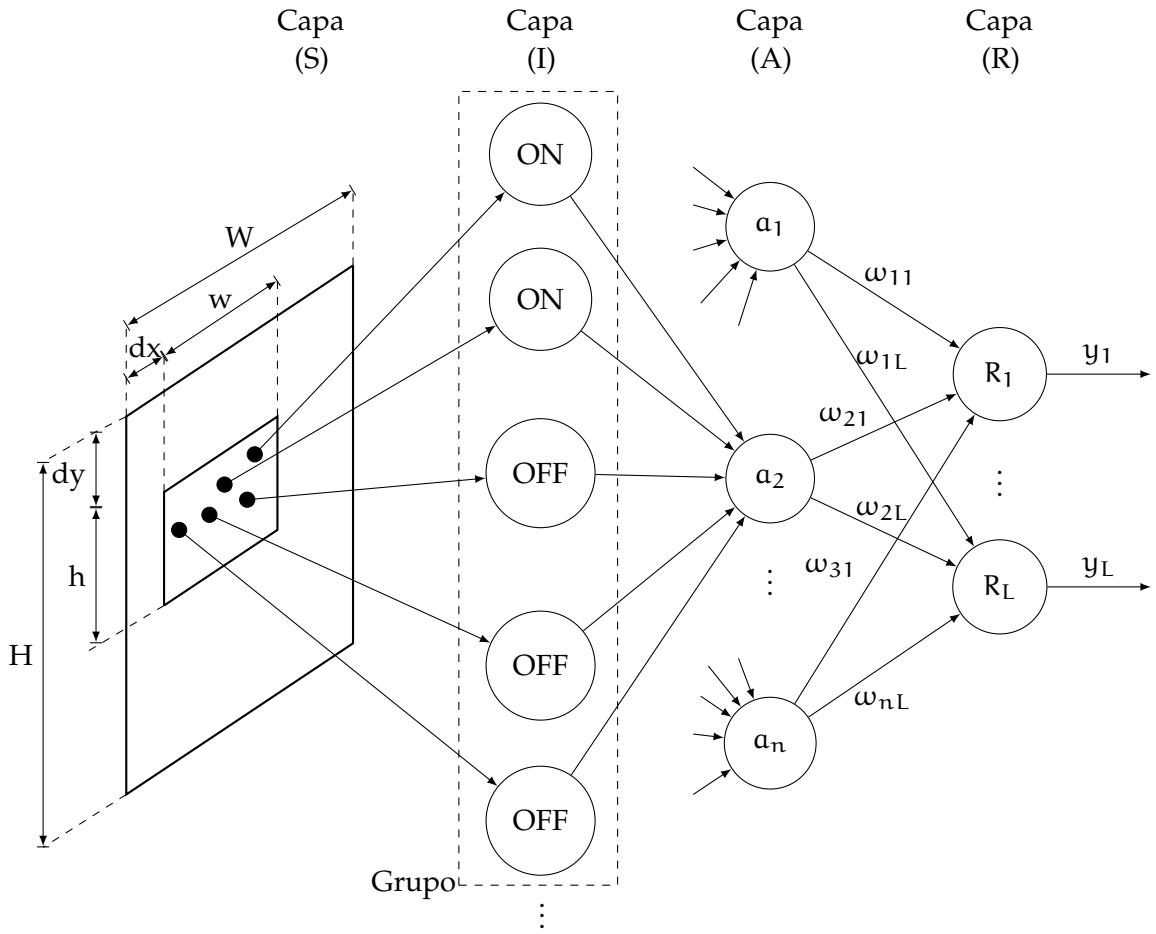


Figura 41: Estructura del clasificador neuronal LIRA-escala de grises[44][7]

tro de la ventana (HxW) se generan de forma aleatoria muchas sub-ventanas (hxw), la posición de la ventana (hxw) es determinada por las coordenadas de la esquina superior izquierda de la ventana, su rango es [0, W-w] y [0, H-h]. Posteriormente, dentro del área (hxw) se seleccionan al azar M neuronas (pixeles de la imagen de entrada) y se designan como como positivos y negativos. Cada una de estas neuronas M están conectadas con una neurona de la capa (I).

CAPA INTERMEDIA (I): La adición de la capa intermedia (I) es una de las modificaciones realizadas al perceptrón de Rosenblatt; cada neurona de la capa (I) tiene conexiones de entrada con la capa (S), y conexiones de salida con las neuronas de la capa (A). Las neuronas en la capa (I) son neuronas ON y neuronas OFF, en la figura 41 (capa I) se muestran dos neuronas ON y tres neuronas OFF, las neuronas ON y OFF están conectadas con los puntos positivos y negativos de la capa (S) respectivamente. Todas las neuronas de la capa (I) que se conectan a **una** neurona de la capa (A) forman un **grupo** de dicha neurona de A; las neuronas ON y OFF tienen el siguiente comportamiento:

$$\Psi_{ON}(x_i) = \begin{cases} 1, x_i \geqslant \theta_i \\ 0, x_i < T_i \end{cases}$$

$$\Psi_{OFF}(x_j) = \begin{cases} 1, x_j \leqslant \theta_j \\ 0, x_j > T_j \end{cases}$$

Donde Ψ es el valor de salida de las neurona ON/OFF, (x_i/x_j) son los valores de entrada de las neuronas ON/OFF y (θ) es su valor umbral, la salida de las neuronas ON es 1 cuando su valor de entrada es mayor que el umbral θ_i, de lo contrario es igual a 0, la salida de las neuronas OFF es 1 cuando su entrada es menor que el umbral θ_j, de lo contrario es igual a 0. Mediante este procedimiento las neuronas de la capa intermedia permiten extraer las características de la imagen (extractor de características intrínseco).

CAPA ASOCIATIVA (A): Corresponde al subsistema de codificación de la imagen, cada neurona de la capa (A) estará activa cuando todas sus conexiones de entrada provenientes de la capa (I) estén activas, es decir, cada neurona a_i se comporta como una compuerta lógica AND, si alguna neurona ONN u OFF no tiene respuesta $(\Psi = 0)$, la neurona a_i tendrá una salida igual a 0 (cero). Por lo tanto, el número de neuronas activas (m) en la capa asociativa será muchas veces menor que el número total de neuronas (n) en esta capa. Se puede utilizar la expresión $m = c\sqrt{n}$ para denotar la relación anterior la cual corresponde a echos neurofisiológicos, donde c es una constante que va de 1 a 5.

El número de neuronas en la capa asociativa puede varias de decenas de miles a varias cientos de miles. Una gran cantidad de neuronas en la capa asociativa permite mejorar la tasa de reconocimiento [44].

CAPA DE REPUESTA (R): Corresponde a la salida de todo el sistema, cada neurona de esta capa corresponde a una de las clases a reconocer, por ejemplo en las tareas de reconocimiento de cifras escritas a mano esta capa contiene 10 neuronas en la capa R, que corresponden a los dígitos del 0 al 9. Las conexiones entre las neuronas de la capa (A) y la capa (R) es: *todas las neuronas conectan con todas las neuronas*, así que las neuronas de la capa (A) se conectan con cada una de las neuronas de la capa (R) y estas conexiones tienen sus respectivos pesos, los cuales son modificados durante el proceso de entrenamiento supervisado utilizando la regla de Hebb. Las excitaciones de cada neurona de la capa de respuesta son calculadas de la siguiente manera:

$$y_i = \sum_{L}^{k=1} \omega_{ki} a_k$$

Y la clase reconocida es definida mediante la neurona de la capa (R) que posea la excitación máxima.

5

IMPLEMENTACIÓN DE REDES NEURONALES ARTIFICIALES

En este capítulo se discuten varios aspectos para la implementación de redes neuronales. Un punto especial es el aprovechamiento inherente del paralelismo de las ANN; se hace una comparación en cuanto a las ventajas y desventajas de la implementación en software y en hardware, así como un análisis de los diferentes sistemas digitales disponibles. También, se describe brevemente la estructura de los FPGA y sus principales familias. Además, se presentan los problemas y restricciones en cuanto a implementar ANN en FPGA. Para finalizar el capítulo, se explica la metodología FPNA/FPNN propuesta para este trabajo.

Las ANN son consideradas poderosos modelos computacionales de naturaleza paralela. No obstante, a pesar de que estos sistemas neuronales están muy simplificados en comparación con los sistemas neuronales biológicos, los cuales poseen un gran numero de sinapsis (ver tab. 10 de la §4.3.6), el número de operadores necesarios y la complejidad de las conexiones en los modelos estándar no son fácilmente manejados por los dispositivos actuales de hardware digital.

Para las redes neuronales, la velocidad de funcionamiento del sistema depende de la forma en como la red es *realizada*, cuando una red neuronal se realiza como un programa que simula la actividad neuronal[1], i.e. en una computadora de propósito general con un lenguaje de alto nivel tal como: JAVA, Python, C#, etc, el rendimiento se ve limitado debido a su naturaleza secuencial que se contrapone a la naturaleza paralela de las redes neuronales. Por eso mismo, es lógico concluir que las redes neuronales artificiales realizadas como programas en computadoras de uso general tendrán un menor desempeño que su implementación equivalente en hardware[2]. Cabe destacar, que las redes neuronales biológicas superan en velocidad y complejidad a las dos anteriores.

En la figura 42 se muestra un gráfico donde se ubica a las redes neuronales artificiales y los sistemas neuronales de algunos organismos vivos; se puede observar como es la relación entre la complejidad estructural del sistema y la velocidad de funcionamiento.

Las redes neuronales típicas realizadas como programas para computadoras de propósito general (redes implementadas en software) de gran popularidad actual, aparecen del lado inferior izquierdo, con los niveles más bajos de complejidad y velocidad de funcionamiento. Por otro lado, se vuelve evidente que es posible lo-

1 Se usará el término *redes neuronales implementadas en software* para hacer referencia a todos aquellos programas secuenciales en lenguaje de alto nivel que simulan la actividad neuronal.

2 Se usará el término *redes neuronales implementadas en hardware* para hacer referencia a que son realizadas mediante métodos de descripción de hardware

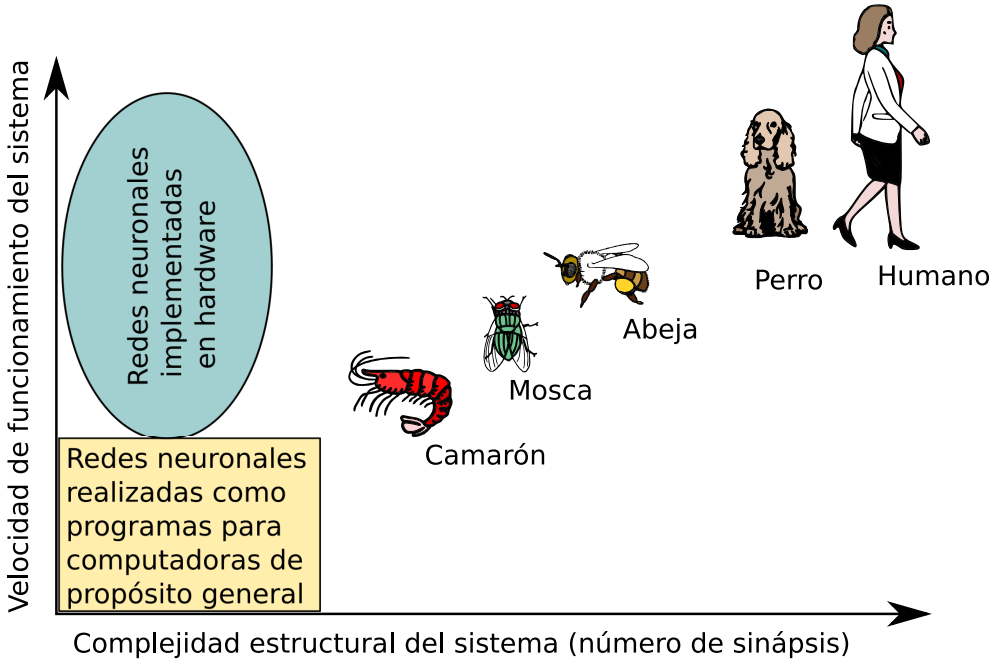


Figura 42: Relación de complejidad y velocidad de funcionamiento[77].

grar una alta velocidad de funcionamiento cuando la red neuronal se realiza en hardware (elipse superior izquierda).

De hecho, una regla general que se deriva del funcionamiento de los sistemas neuronales biológicos, es que: mientras mayor sea el número de elementos de un sistema (neuronas y sinapsis) y todos esos elementos trabajen juntos simultáneamente (de forma paralela), la velocidad de procesamiento de los datos aumentará proporcionalmente a las dimensiones estructurales de dicho sistema[**82**].

Existen numerosos ejemplos de redes neuronales realizadas en hardware como chips electrónicos especializados (ASIC) o mediante matrices de compuertas programables en campo (FPGA), otras soluciones involucran el uso de optoelectrónica o incluso *neurochips* que combinan tanto silicio electrónico como partes biológicas (células neuronales vivas o tejidos neuronales tratados) como componentes del dispositivo.

Todos estos métodos de realización de redes neuronales en hardware pueden operar muy rápido; no obstante, la complejidad estructural de las redes limita a veces el uso de estas tecnologías. Así mismo, la flexibilidad para aplicar estos sistemas en hardware no siempre es satisfactoria para la mayoría de los usuarios, por lo que es comprensible que en la práctica la mayoría de los desarrolladores de redes prefieran soluciones de software a pesar de sus limitaciones. Sin embargo, no debe perderse de vista la naturaleza paralela de grano fino[3] que es propia de las redes neuronales, la cual utiliza muchos intercambios de información y por lo tanto siempre se adaptaran mejor cuando son implementan en hardware.

3 Referido al concepto de granularidad en computación paralela.

COMPUTADORAS PARALELAS DE PROPÓSITO GENERAL: Las implementaciones de paralelismo de grano fino en computadoras masivamente paralelas, ya sean del tipo: una instrucción, múltiples datos (SIMD) o múltiples instrucciones, multiples datos (MIMD), sufren de conectividad en los modelos neuronales estándar, lo que resulta en intercambios de información poco eficientes. Por otro lado, el paralelismo de grano grueso se aplica principalmente al aprendizaje neuronal, y su eficiencia sufre de la secuencialidad de los algoritmos de aprendizaje estándar. Además, las computadoras masivamente paralelas son recursos caros y no se pueden usar en aplicaciones integradas, dichas soluciones suelen ser preferibles sobretodo para estructuras neuronales enormes y cálculos neuronales complejos así como métodos de aprendizaje.

5.1 Hardware para las redes neuronales

Durante la década de 1980 y a principios de 1990, se realizaron grandes esfuerzos, tanto en el área académica como industrial, para el diseño e implementación de neurocomputadoras de hardware. Sin embargo, hasta el día de hoy no han sido utilizadas ampliamente[4]; esto puede atribuirse al hecho de que la tecnología empleada fue basada en su mayoría en circuitos integrados de aplicación específica (ASIC, por sus siglas en inglés), pero nunca fue lo suficientemente desarrollada o competitiva como para justificar un desarrollo a gran escala. Así mismo, los arreglos de compuertas lógicas que se desarrollaron en ese período nunca fueron lo suficientemente numerosos ni rápidos para su uso en aplicaciones serias de redes neuronales[60].

En la actualidad existe una amplia familia de sistemas digitales que provee de opciones a los diseñadores para seleccionar una plataforma de hardware en la cual implementar las ANN (la fig. 43 muestra un panorama general). No obstante, no todas las tecnologías son adecuadas ni se adaptan bien a las necesidades de las redes.

Debido a la gran cantidad de operaciones aritméticas requeridas por las aplicaciones con redes neuronales ha sido necesario desarrollar métodos de implementacion rápida, para dichas implementaciones se han utilizado varios tipos de dispositivos paralelos tales como computadoras masivamente paralelas, neurocomputadoras, ASIC analógicos o digitales y los FPGA. A continuación se describen brevemente algunas características que relacionan estas tecnologías con las redes neuronales[60]:

COMPUTADORAS PARALELAS DEDICADAS: Los sistemas paralelos dedicados a la computación neuronal son llamados neurocomputadoras, están basados en dispositivos informáticos como los DSPs o los neuroprocesadores; su uso esta limitado por su costo y el tiempo de desarrollo necesario. Ademas, se vuelven obsoletos rápidamente, en comparación con los procesadores secuenciales más recientes.

ASIC ANALÓGICOS: Son muy rápidos, densos y de baja potencia, por lo que se han realizado muchas implementaciones de redes con ellos; no obstante, presentan problemas específicos de precisión, almacenamiento de datos y robustez, el apren-

4 Cabe mencionar que todo el campo ya estaba en decadencia a finales de la década de 1990[60]

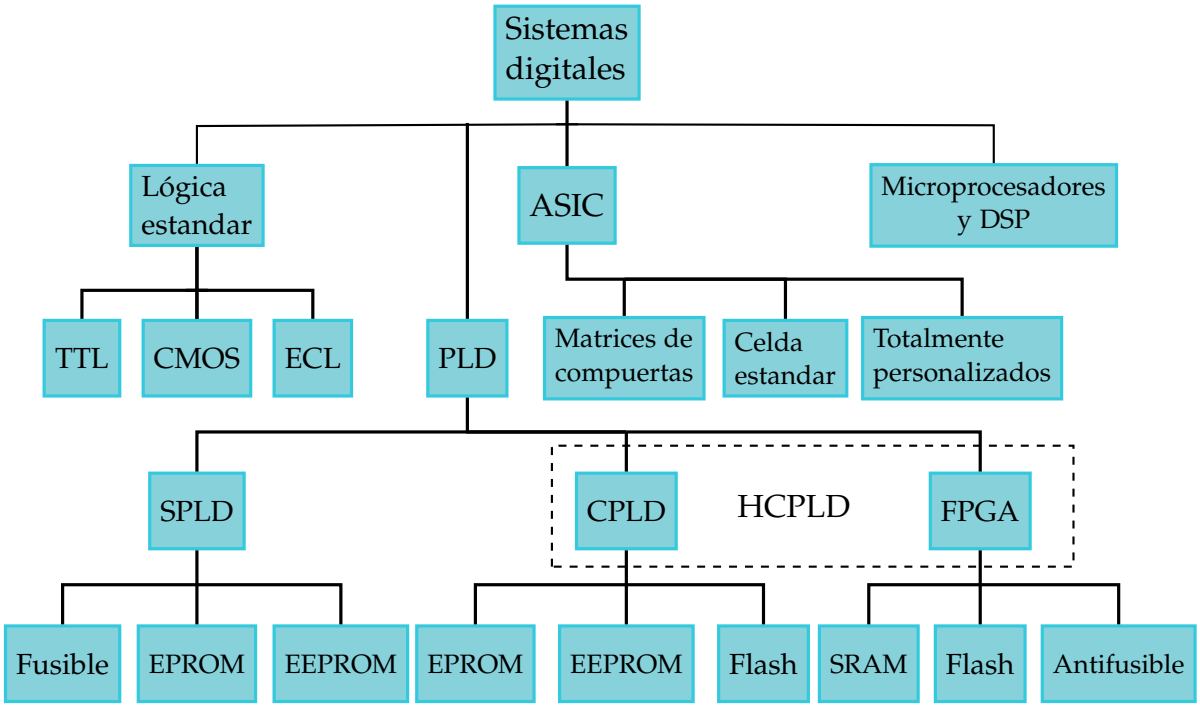


Figura 43: Árbol familiar de los sistemas digitales. [78]

dizaje en chip es difícil y se consideran una solución costosa y para nada flexible. Ademas de esto, el desarrollo es largo y complicado sobretodo para usuarios que no son expertos en tecnologías analógicas.

ASIC DIGITALES: En comparación con los chips analógicos, proporcionan más precisión, son más robustos y pueden manejar cualquier cálculo neuronal estándar; Sin embargo, su diseño requiere un gran esfuerzo para seleccionar los chips de trabajo y resulta ser costoso cuando solo se necesita fabricar lotes pequeños. Por lo general, se implementan partes limitadas de redes neuronales en ellos, para después ser incluidas en algún sistemas de neurocomputadoras.

FPGA: Con los dispositivos de hardware programables se da solución a muchos de los problemas mencionados anteriormente, los dispositivos FPGA ofrecen un combinación entre la eficiencia del hardware de los ASIC digitales y la flexibilidad de un manejo simple similar al del software. Sin embargo, la implementación de modelos neuronales estándar en FPGA plantea problemas específicos, principalmente relacionados con los operadores utilizados y las topologías complejas.
En los años 80t's, los primeros FPGA eran pequeños en términos de recursos lógicos, por lo que tendían a usarse principalmente para proporcionar una interconexión flexible, interfaz lógica (a veces llamada lógica de pegamento), y para implementar controladores basados en máquinas de estados finitas. Se requerían múltiples FPGA para implementar cualquier sistema de procesamiento de imágenes significativo. Sin embargo, a medida que los FPGA crecían en capacidad, se desarrollaron sistemas híbridos en los que los FPGA se usaban como coprocesadores de visión o aceleradores dentro de una computadora host. No obstante, actualmente los FPGA tienen recursos suficientes para permitir la implementación de aplicacio-

nes completas en un solo FPGA, lo que los convierte en una opción ideal para los sistemas integrados de visión en tiempo real[5].

5.2 Estructura FPGA

Un FPGA es un circuito integrado que consiste en una matriz de bloques lógicos programables y un enrutamiento programable entre estos bloques, lo que permite que el dispositivo se configure para realizar funciones lógicas digitales complejas.

Definición 5.2.1. Definición de FPGA: Una matriz de compuertas lógicas programable en campo **FPGA,por sus siglas en inglés!**, es un circuito integrado diseñado para ser configurado por un cliente o un diseñador posterior a su fabricación, por consiguiente, *programable en campo*.

Una FPGA consta de tres partes principales:

1. Un conjunto de celdas lógicas programables también llamadas bloques lógicos o bloques configurables
2. Una red de interconexión programable
3. Un conjunto de celdas de entrada y salida

La configuración de un FPGA generalmente se especifica utilizando algún lenguaje de descripción de hardware (HDL), similar a los utilizados para un ASIC. Los FPGA se inventaron para proporcionar un dispositivo digital de propósito general con gran flexibilidad y utilidad. La particularidad de los FPGA es que en lugar de implementar la lógica mediante compuertas de transistores, se implementa mediante memoria, diseñado como tablas de búsqueda (LUT, por sus siglas en inglés). Los FPGA utilizan estas tablas de búsqueda como elemento lógico base ya que los LUT son bastante versátiles para el diseño lógico. Conceptualmente una LUT puede entenderse como un conjunto de celdas de memoria programables cuya salida se selecciona mediante un multiplexor, lo que permite la selección de una celda de memoria específica.

En la figura 44 se muestra un esquema simplificado de los elementos fundamentales de una celda lógica típica de FPGA; consta de una LUT de 4 entradas, un sumador completo (FA, por sus siglas en inglés), y un flip-flop tipo D; los LUT de esta figura se dividen en dos LUT de 3 entradas. En modo normal, las entradas se combinan en una LUT de 4 entradas a través del multiplexor izquierdo., mientras que en el modo aritmético, las salidas se envían al FA; la selección del modo se programa en el multiplexor central.
El tipo de memoria utilizado para crear los conmutadores de enrutamiento en los FPGAt's, es el aspecto más importante para determinar las diferencias en el desempeño y sus capacidades. Los FPGA antifusible son conocidos por su confiabilidad, pero son costosos y programables una sola vez (OTP, por sus siglas en inglés), los FPGA con memoria FLASH también son muy fiables y son reprogramables, pero son más costosos que los que utilizan memoria SRAM, los cuales son programables y tienen la mayor densidad y el menor costo para una lógica equivalente.

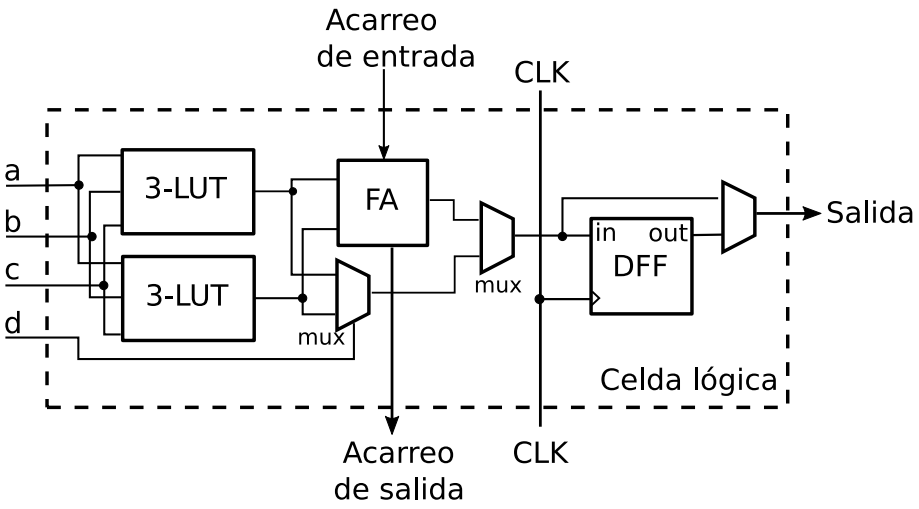


Figura 44: Celda lógica FPGA

En la tabla 12 se presentan los cuatro principales proveedores de dispositivos lógicos programables, los cuales tienen una variedad de opciones que incluyen muchos tipos de configuración.

Tabla 12: Variedad de configuraciones por proveedor

Producto	Altera	Xilinx	Lattice	Microsemi
FPGA antifusible				○
FPGA FLASH		○		○
FPGA SRAM	○	○	○	

En los dispositivos Xilinx, los bloques lógicos se denominan Bloques lógicos configurables (CLB, por sus siglas en inglés), un CLB proporciona elementos funcionales para la lógica combinacional y secuencial, incluidos los elementos básicos de almacenamiento. El número exacto y las características varían de un dispositivo a otro, pero cada CLB consta de una matriz de interruptores configurables con 4 o 6 entradas, así como circuitos de selección y flip-flops. En los FPGA Xilinx, los CLB se pueden conectar de manera eficiente a los CLB vecinos, así como a los CLB de la misma fila o columna; la estructura de comunicación configurable puede conectar CLB externos a bloques de entrada/salida (IOB) que controlan los pads de entrada o salida del chip.

El corazón de toda computadora digital es la unidad central de procesamiento (CPU), el corazón de cada CPU es la unidad lógica aritmética (ALU), y el corazón de cada ALU es el circuito sumador. Los sumadores son fundamentales, una vez que se puede sumar, se puede restar agregando un complemento y también se puede multiplicar contando cuántas veces suma. Por lo anterior, no es de extrañar que los FPGA estén infundidos con estructuras para facilitar el diseño y mejorar el rendimiento de los sumadores.

En cuanto a diseñar multiplicadores en FPGA, el multiplicar dos números es una operación lógica común; no obstante, que puede requerir una cantidad conside-

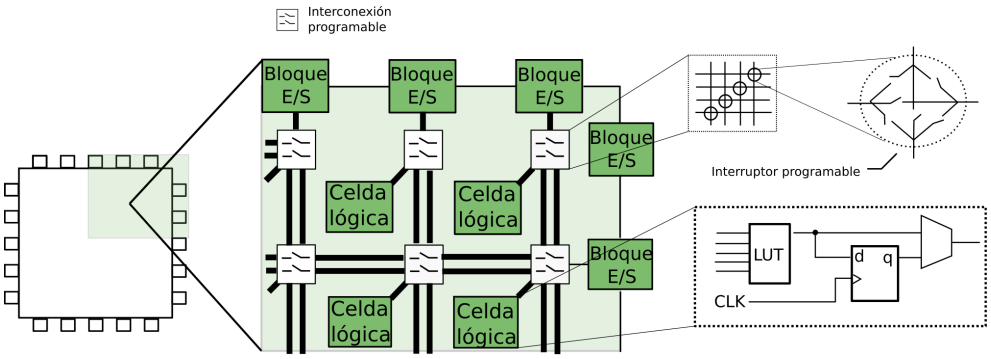


Figura 45: Estructura conceptual de un dispositivo FPGA

rable de circuitos, esto ha sido un problema para los diseñadores durante años. Afortunadamente, existen varias formas de implementar circuitos multiplicadores en FPGA:

1. Circuitos combinacionales
2. Cambio secuencial y adición (sequential shift and add)
3. Algoritmo especializados, como el algoritmo de Booth,, el multiplicador Dadda o el multiplicador de árboles de Wallace
4. Memorias (look up tables)
5. Combinación de las anteriores
6. Bloques multiplicadores duros (HMB), construidos dentro del FPGA

Mediante circuitos combinacionales se consigue rapidez pero son grandes, El cambio secuencial y adición tiene un enfoque de máquina de estados que es pequeño pero lento, los algoritmos de estados por otro lado, son rápidos y pequeños pero complejos y las tablas de búsqueda son una buena solución, son simples pero no son escalables del todo. Por lo que, si están disponibles, los HMB es la mejor solución en la mayoría de los casos. Los multiplicadores duros, proporcionan la mayor velocidad de todas las implementaciones antes mencionadas,, funcionan a 200 MHz o más, es decir, 50 nanosegundos por multiplicación.

5.3 Diseño en FPGA

Un diseño en FPGA primero requiere de la descripción de varios bloques operativos y luego agregar el control y los esquemas de comunicación a la descripción. Posteriormente, una herramienta automática de *compilación* proyecta el circuito descrito en el chip. El hardware configurable ha demostrado estar bien adaptado para desarrollar implementaciones de redes neuronales flexibles y eficientes.

Cuando se diseña utilizando un lenguaje de descripción de hardware (HDL), las expresiones y funciones lógicas deben asignarse a los bloques lógicos de bajo nivel en un FPGA. Para hacer esto, es necesario llevar a cabo tres funciones específicas:

1. Asignación: Asignar funciones lógicas a los CLB.

2. Colocación: Coloca los CLB en el FPGA.

3. Enrutamiento: Realizar las conexiones entre CLB

Actualmente los ordenadores ofrecen herramientas especiales para la creación y verificación de diseños. Esto, unido a la metodología Top-down de diseño de circuitos, fomentó la aparición de herramientas de descripción que permiten al diseñador definir un problema de forma abstracta[**Pardo1997**]. Teniendo en cuenta esta evolución, las herramientas de CAD actuales permiten realizar la descripción de una idea o diseño electrónico de las siguientes formas:

- **Descripción estructural:** Basa su comportamiento en modelos logicos ya establecidos (compuertas, sumadores, etc), estas estructuras pueden ser diseñadas por el usuario. Dependiendo de la herramienta a utilizar, existen dos formas de realizarla:
  - **Mediante esquemas:** Consiste en la descripción gráfica

    de los componentes de un circuito, puede decirse que es la forma tradicional de diseñar circuitos.
  - **Mediante lenguaje:** Se realiza una enumeración de los componentes de un circuito así como sus conexiones.
- **Descripción comportamental:** Se describe un circuito electrónico (generalmente digital) simplemente describiendo cómo se comporta. Se utiliza un lenguaje de descripción hardware específico.
  - **Por flujo de datos:** Muestra en detalle la transferencia de información entre las entradas y las salidas de una entidad.
  - **Descripción Funcional:** Lo más importante es el conocimiento global del sistema, las entidades son diseñadas como una caja negra.

5.3.1 *Consumo de potencia*

En todas las aplicaciones integradas, y en particular de la FPGA, el consumo de energía del sistema es de gran importancia. La potencia disipada por un FPGA se puede dividir en dos componentes: potencia estática y potencia dinámica; la *potencia dinámica* es la potencia necesaria para cambiar las señales de 0 a 1 y viceversa, por el contrario, la potencia estática es la que se consume cuando no se cambia, existe simplemente por el hecho de que el dispositivo esté encendido.

La mayoría de los FPGA se basan en tecnología CMOS; el circuito CMOS más básico se muestra en la Figura 46; consiste en un transistor pull-up de canal P y un transistor pull-down de canal N. Solo uno de los transistores está encendido a la vez, y dado que los transistores CMOS tienen una resistencia R_{dsoff} muy alta, los circuitos MOS generalmente tienen una disipación de potencia estática muy baja[5].

Por lo tanto, cualquier potencia significativa solo se disipará cuando la salida cambia de estado; esto resulta de dos fuentes principales[**sawar1997**]. La primera es al cargar la capacitancia en la línea de salida; la capacitancia resulta de las entradas

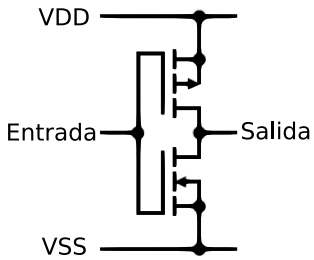


Figura 46: Inversor CMOS

de cualquier puerta conectada a la salida, combinada con la capacitancia parásita asociada con el cableado entre la salida y cualquier entrada de menor potencial. La potencia requerida viene dada por:

$$P = NCV_{DD}^2 f \tag{5.3}$$

donde N es el número promedio de salidas que cambian en cada ciclo de reloj, C es la capacitancia promedio en cada salida, f es la frecuencia de reloj y V_{DD} es el voltaje positivo de alimentación. Para un sistema integrado, el consumo de energía puede reducirse considerando cada uno de los términos de la ecuación anterior[5]:

- Limitando el número de salidas que cambian con cada ciclo de reloj
- Minimizando el fan-out de cada puerta y minimizando el uso de cables largos (para mantener la capacitancia baja)
- Reduciendo el voltaje de la fuente de alimentación (aunque el desarrollador tiene poco control sobre esto, se ha reducido con cada generación tecnológica)
- Reduciendo la frecuencia del reloj.

La reducción de la frecuencia del reloj tiene el efecto más significativo en la reducción del consumo de energía, ya que a menudo es la señal del reloj la que disipa la mayor cantidad de energía, ya que se alterna con cada ciclo; y debido a que la red de distribución del reloj se extiende por todo el FPGA, también tiene una alta capacitancia.

El segundo componente de la disipación dinámica de potencia se debe al hecho de que cuando un transistor se apaga, el otro se enciende; por lo tanteo, durante la transición hay un breve intervalo de tiempo donde ambos transistores están parcialmente encendidos y existe una ruta de conducción entre VDD y VSS. Para velocidades de transición rápidas, la disipación de energía de esta fuente suele ser mucho menor que la de cargar la capacidad de carga[5].

La disipación de potencia estática surge de las corrientes de fuga de cinco fuentes diferentes de las cuales predomina la fuga por debajo del umbral[**unsal2003**]; esta es la corriente de fuga a través del transistor cuando el voltaje de la puerta está por debajo del umbral de conmutación y el transistor debe estar apagado. A medida que se reduce el voltaje de suministro de los dispositivos, el voltaje de umbral también debe reducirse para mantener la velocidad del dispositivo. No obstante, la

corriente de fuga por debajo del umbral aumenta exponencialmente al disminuir el voltaje de umbral, por lo que la corriente de fuga contribuye cada vez más a la cantidad de potencia disipada total con generación de tecnológica. [**ka02002**, **unsal2003**].

5.3.2 *Canalización de bajo nivel*

En la mayoría de las aplicaciones no triviales, los retardos en la propagación de datos a través de la lógica, necesaria para implementar todas las operaciones de procesamiento de imágenes, excede la frecuencia de reloj. La canal canalización o pipelining, como se le conoce en inglés, divide la lógica en bloques más pequeños distribuidos en varios ciclos de reloj, lo que reduce el retraso de propagación en cualquier ciclo de reloj. Esto, permite mejorar el rendimiento y cumplir con las restricciones de tiempo. No obstante, se debe tener en cuenta que la latencia del sistema aumentará.

5.3.3 *Rendimiento*

Existen como mínimo dos cuestiones que deben considerarse para una evaluación adecuada del rendimiento: las métricas utilizadas y qué puntos de referencia son utilizan para obtener dichas mediciones; ambos deben ser tales que, en general, puedan ser fácilmente acordados y entendidos por la mayoría de los usuarios. Por desgracia, el área de la redes neuronales carece de ambos; las métricas más utilizadas son las conexiones por segundo (CPS, por sus siglas en inglés), que es esencialmente la velocidad a la que se llevan a cabo las múltiples operaciones de adición de las neuronas, y las actualizaciones de las conexiones por segundo (CUPS,por sus siglas en inglés) que es esencialmente la velocidad a la que se realizan las actualizaciones de los pesos sinápticos. La aceleración también es usada a veces, pero tiene incluso menor valor que las CPS y CUPS. Como es el caso de las computadoras de uso general, en última instancia, la mejor medida del rendimiento es el tiempo de ejecución real[60].

5.3.4 *Verificación*

5.3.4.1 *Análisis de tiempo:*

Los relojes son fundamentales para la electrónica digital moderna, proporcionan sincronización la cual es necesaria para transmitir datos sin errores; A medida que los datos de sistema digital pasan de un valor a otro, durante el tiempo de transición, el valor de los datos será incorrecto, tenemos que esperar a que los datos estén estables antes de usarlos y la señal de Sincronización es el reloj.

Los datos deben ser estables antes y después del flanco del reloj para que puedan ser transferidos de forma fiable. Si no se sincroniza correctamente, habrá una serie de problemas como:

- Hazards, en inglés Señales inesperadas o no deseadas

- Metaestabilidad, una condición causada por violaciones de tiempo y salidas de flip-flop que conducen a datos erróneos
- Race conditions, que conducen a violaciones que ocurren dependiendo de las condiciones de operación, errores que van y vienen
- Sesgo de reloj, otra fuente inesperada de violaciones

Los peligros estáticos (Hazards) ocurren como consecuencia de retrasos y por lógica desigual, la mejor forma de eliminarlos es utilizando flip flops y un diseño síncrono, que permite la transferencia de datos solo en un flanco de reloj cuando los datos son correctos y estables.
La revisión de los tiempos setup y hold es uno de los tipos comprobación más comunes para la verificación de tiempos. Las entradas síncrona tienen especificaciones para los tiempos setup y hold.

TIEMPO SETUP: se define como la cantidad mínima de tiempo, antes del un flanco activo del reloj, en que los datos deben estar estables para que se transfieran correctamente. Dicho de otro modo, cada flip-flop (o cualquier elemento secuencial, en general) necesita algo de tiempo para que los datos permanezcan estables antes de que llegue el flanco del reloj, de modo que pueda capturar los datos de manera confiable.

TIEMPO HOLD: se define como la cantidad mínima de tiempo después del flanco activo del reloj durante el cual los datos deben ser estables. De manera similar al tiempo setup, cada elemento secuencial necesita algo de tiempo para que los datos se mantengan estables después de que llegue el borde del reloj para capturar datos de manera confiable.

Para que el chip funcione correctamente, las restricciones de tiempo de configuración y retención deben cumplirse adecuadamente para todos y cada uno de los flip-flop del diseño. Si existe incluso un solo flop que no cumpla con los requisitos de setup y hold el diseño fallará y habrá metaestabilidad.

5.3.5 *Memoria*

En las primeras arquitecturas FPGA, los flip-flops que forman parte de los bloques lógicos eran los únicos elementos de memoria disponibles. Por lo tanto, era difícil almacenar una gran cantidad de datos dentro del chip FPGA y era necesario utilizar una memoria externa para ciertas aplicaciones. No obstante, en esta configuración, el ancho de banda entre el FPGA y la memoria externa a menudo tiende a convertirse en un cuello de botella y perjudica el rendimiento. Actualmente en los FPGA comerciales, se ha logrado implementar eficientemente elementos de memoria dentro del chip, esta memoria se conoce comúnmente como memoria embebida, y se clasifica generalmente en dos tipos[**Aman02018**]:

1. Bloques de memoria como macros de hardware (módulo de RAM embebida)
2. Utilizando las LUT de los bloques lógicos (RAM distribuida)

Introducir bloques de memoria en chip como macros de hardware es un enfoque que proporciona una funcionalidad de memoria eficiente en FPGA, en las arquitecturas FPGA de Xilinx, estos bloques de memoria se denominan *Block RAM* (BRAM). Los BRAM no solo se pueden utilizar como memoria de puerto único, sino también como memoria de puerto dual, esta propiedad permite a los usuarios configurar fácilmente memorias de tipo FIFO[**Amano2018**].

Algunas operaciones de procesamiento de imágenes requieren que las imágenes se almacenen parcial o totalmente en un búfer. Si bien los dispositivos FPGA actuales de alta capacidad tienen suficiente memoria en el chip para almacenar una imagen, se considera, en la mayoría de las aplicaciones, que es un mal uso de los recursos utilizarlos simplemente como un búfer de imágenes. Por esta razón, los fotogramas de imágenes y otros conjuntos de datos grandes se almacenan con mayor frecuencia en una memoria fuera del chip. Esto por supuesto coloca grandes cantidades de datos detrás de un ancho de banda limitado y conexiones serializadas[5].

5.3.6 *Familias FPGA*

Los dos principales fabricantes de FPGA, en términos de participación de mercado, son Xilinx y Altera, aunque también existen otros tales como Lattice, Microsemi y Atmel.

Xilinx fue uno de los primeros desarrolladores de tecnología para arreglos de compuertas programables en campo. Ha tenido varias series de de dispositivos y familias a lo largo del tiempo, los más conocidos son los pequeños FPGA $SPARTAN-6$, y FPGA más grandes como el $ARTIX-7$, $VIRTEX-7$ y $KINTEX-7$. No obstante, hoy en día Xilinx cuenta con familias con mayor capacidad como la familia *Ultra Scale* y *Ultra Scale+*, cada una con dispositivos KINTEX y VIRTEX. La principal diferencia entre estas familias son las proporciones de sus distintos recursos (LUT, BRAM, etc), así como la velocidad de operación; los dispositivos Virtex están diseñados principalmente para un alto rendimiento, mientras que los dispositivos Kintex buscan brindar el mejor precio/rendimiento.

Por otro lado, Altera ofrece dispositivos pequeños de la serie Cyclone y MAX, la serie ARRIA de rango medio y la serie STRATIX de alto rendimiento. Así mismo, Altera ofrece su familia más reciente AGILEX construida con tecnología de proceso de 10 nm, la cual ofrece hasta un 40 % más de rendimiento e integran el procesador Arm*Cortex-A53 de cuatro núcleos para proporcionar una alta integración del sistema.

La empresa Microsemi ofrece dispositivos FPGA con tecnología Flash como lo son IGLOO e IGLOO2, los cuales son soluciones rápidas de un solo chip reprogramables, con un buen rendimiento, bajo consumo y alta fiabilidad y seguridad antifusibl. También, ofrece FPGA antifusible que están diseñados para entornos robustos y alta confiabilidad, incluyendo la familia ACCELERATOR y la familia tolerante a la radiación RTAX. Si la fiabilidad, seguridad o la baja potencia son requisitos de diseño, los FPGA Microsemi merecen ser considerados

5.3.7 *Placas de desarrollo*

En la actualidad, existe una amplia gama de kits de desarrollo para todos los niveles de presupuesto y requisitos de rendimiento de los propios fabricantes, o de compañías externas especializadas en kits de desarrollo; los fabricantes de FPGA son proactivos en el suministro de software de diseño mediante la web (a menudo de forma gratuita para fines no comerciales), lo que ha permitido que no existan obstáculos para que los ingenieros obtengan acceso tanto a las herramientas de diseño como al hardware con el cual probar sus conceptos.

Muchos fabricantes de FPGA también venden placas de desarrollo o placas de evaluación. estas placas a menudo consisten en un FPGA con un conjunto de periféricos para demostrar las capacidades del dispositivo dentro de una gama de aplicaciones específicas. En particular, la mayoría de las placas incluyen algún tipo de memoria externa y algunos medios para interactuar con una computadora host (incluso si es solo para descargar archivos de configuración). Las placas destinadas al uso educativo tienden a ser de menor costo, a menudo con un FPGA más pequeño y una amplia gama de periféricos, lo que permite que se utilicen para la docencia en el laboratorio.

Criterios para elegir un FPGA

El objetivo en la selección de dispositivos lógicos programables, es elegir la pieza que mejor se ajuste a nuestros requisitos, para ello se utilizaron once criterios de selección de dispositivos lógicos programables par evaluar a estos dispositivos, los cuales se muestran a continuación:

1. Tamaño o densidad lógica (cantidad de lógica en las compuertas del sistema, LEs, Slices, ALM, etc.)
2. Costo por compuerta lógica
3. Velocidad (frecuencia de reloj máxima)
4. Consumo de energía (estático y dinámico)
5. Reprogramabilidad (el tipo de memoria de configuración)
6. Costo por E/S (densidad de E/S)
7. IP de hardware disponible en chip (memoria, bloques DSP, transceptores, etc)
8. Sincronización determinista (el tiempo es consistente en cada implementación)
9. Fiabilidad (medido por la tasa FIT)
10. Resistencia (número de ciclos e programación y años de retención)
11. Diseño y seguridad de datos

5.4 Taxonomía de la implementación de ANNs en FPGA

Las ANN son redes paralelas y distribuidas de unidades de procesamiento no lineales simples, interconectadas en una disposición en capas; las características computacionales típicamente asociadas con ellas son:

1. El paralelismo
2. La modularidad
3. La adaptación dinámica

Los FPGA poseen varias ventajas especificas relacionadas con la implementación de redes neuronales, se destacan como dispositivos de hardware apropiados los cuales concilian topologías de hardware simples con arquitecturas neuronales complejas, lo cual logran utilizando algunos principios de hardware configurable aplicados al cálculo neuronal. Así mismo, los FPGA ofrecen una opción barata, fácil, robusta y flexible ya que pueden usarse para aplicaciones integradas, incluso para producciones de baja escala [60].

Así mismo en los FPGA es posible explotar su rápida reconfiguración para actualizar los valores de los pesos o incluso la topología de la red. Al utilizar la reconfigurabilidad, actualmente se han podido generar estrategias para implementar redes neuronales en FPGA de forma económica y eficiente[85]; el hardware basado en FPGA SRAM puede reconfigurarse rápidamente un número ilimitado de veces, esta flexibilidad permite la creación rápida de prototipos con diferentes estrategias de implementación, así como algoritmos de aprendizaje, simulaciones iniciales o pruebas de concepto.

Lo anterior es de gran utilidad ya que en la mayoría de las aplicaciones se deben probar varias arquitecturas neuronales para encontrar la más eficiente y esto puede hacerse de manera directa sin ningún costo adicional. Igualmente, una buena arquitectura que ha sido diseñada e implementada en un dispositivo en particular, puede ser reemplazada más tarde por una mejor sin tener que diseñar un nuevo chip. El proyecto GANGLION [12] es un buen ejemplo de creación rápida de prototipos.

El aprendizaje de la red en chip (On-chip) a menudo se considera difícil e imprácticO; de hecho, se usa raramente ya que resulta en una pérdida de eficiencia para la implementación de hardware y requiere una mayor precisión así como de algunos operadores específicos[60]. Por lo tanto, el aprendizaje fuera de chip (off-chip) se elige de manera más natural, sobretodo si la aplicación requiere de un aprendizaje dinámico.

Teniendo en cuenta que la velocidad y el área disponible en un FPGA se duplican aproximadamente cada dos años, la implementación realizada puede asignarse a nuevos y mejorados FPGA, lo cual significa una gran ventaja y plantea la posibilidad de que cada vez sea más común que las redes neuronales muy grandes puedan ser implementarse en FPGA individuales, teniendo en cuenta ,claro, que el método de implementación siempre sea suficientemente escalable.

5.4.1 *Problemas de implementación*

La implementación de redes neuronales en FPGA logra solucionar diferentes problemas que se presentan en otras soluciones de hardware digital, como se explica en [60], tambien otros problemas se agudizan debido a las limitaciones específicas que presentan los FPGA, particularmente las relacionadas con los requerimientos de área y de topología del sistema.

Una de las operaciones aritméticas más importantes llevadas a cabo en las redes neuronales es el cálculo de los productos internos, el cual es necesaria para realizar la suma ponderada (ecuación 4.1). Esto significa, que existen no solo multiplicaciones y adiciones individuales, sino también una alternancia de multiplicaciones y adiciones sucesivas, en otras palabras, se realiza una secuencia de operaciones de multiplicación-suma la cual puede ser fácilmente implementada mediante una maquina de multiplicación acumulación (también conocida como MAC). No obstante, aunque las operaciones aritméticas sean de alguna manera sencillas, la gran cantidad de neuronas presentes en la red (incluso miles) que deben realizar estas operaciones de manera simultanea representan una limitación critica en la tarea de implementación.

Los FPGA tienen un número limitado de bloques lógicos y recursos de enrutamiento, el diseñador tiene que considerar esto y realizar diseños realistas utilizando algún lenguaje de descripción de hardware (HDL, por sus siglas en inglés). El estilo de descripción para el código en HDL, debe hacer el mejor uso de los recursos, aunque el código puede ser transferible entre tecnologías algunas veces puede ser necesario reescribirlo para obtener mejores resultados[83].

PROBLEMAS DE TOPOLOGÍA: Las topologías de los dispositivos de hardware las cuales son en 2D, no se ajustan fácilmente a los grafos de conexión 3D de la mayoría de los modelos neuronales, por ejemplo, los que contienen múltiples capas que aunque presentan regularidad pueden ser muy complejos[60]. Además, aunque los FPGA posen la característica de ser programables, presentan un marco de conexión ya especificado por el diseño particular del dispositivo (número de CLBs, etc.) por lo que su conexión no es totalmente libre. También, existen problema de enrutamiento que no pueden resolverse fácilmente modificando los circuitos.

PROBLEMAS DE ÁREA: Las redes neuronales generalmente no requieren cálculos muy precisos, no obstante, la precisión requerida puede variar con la aplicación, y esta depende en gran medida de los operadores y funciones elementales utilizadas (e.g, la función de activación). Sin embargo, estos pueden ser *codiciosos* en cuanto al área requerida para su implementación, sobretodo si se incluye el proceso de aprendizaje en el chip. El manejo cuidadoso de los accesos a memoria debes ser imprescindible con el fin de lograr una implementación eficiente[60]. Almacenar el valor de los pesos sinápticos en el FPGA también puede ser un problema para redes neuronales grandes.

Surge un inconveniente peculiar en las aplicaciones de redes neuronales, y es que existe un desequilibrio entre el tiempo de cálculo total de la red y el área de im-

plementación de los diferentes operadores involucrados en el proceso, es decir, la mayor parte del tiempo de cálculo se ocupa solo en una pequeña parte de todo el proceso, en particular, para el cálculo de las sumas ponderadas que requieren operaciones de multiplicación y adición. Ademas de esto, otros operadores que consumen gran cantidad de área deben ser implementados (como las funciones de activación).Una parte significativa del área estrictamente limitada de un FPGA es requerida para implementar estos operadores, de modo que al final una parte muy reducida de toda el área disponible ocupa casi el 100 % del tiempo de cálculo total[60].

Se han definido diferentes paradigmas neuro-computacionales que contrarrestan los principales problemas de implementación y conducen a modelos neuronales más tolerantes a las restricciones de hardware mencionadas; dado que los principales problemas de implementación están vinculados a los operadores codiciosos de área y a las topologías complejas, se pueden diferenciar dos tipos principales de paradigmas mutuamente complementarios: los primeros, son marcos de cómputo neuronal para ahorran área y los segundos son paradigmas capaces de gestionar las topologías e incluso simplificarlas.

Algunas soluciones para el ahorro de área serían por ejemplo: la implementación optimizada de la función de activación, la aritmética en serie de bits, o la aritmética basada en pulsos[5]. Mientras que dentro de las soluciones topologías entra el marco de los FPNA, los cuales permiten simplificar las arquitecturas de los modelos neuronales estándar sin una pérdida significativa de su capacidad[60]. La siguiente sección presenta el concepto de Matrices Neuronales Programables en Campo y Matrices Neuronales Programables en Campo (FPNA, por sus siglas en inglés) y Redes Neuronales Programables en Campo (FPNN, por sus siglas en inglés) y se explica cómo estos permiten superar la mayoría de los problemas de implementación mencionados anteriormente.

5.5 FPNAs y FPNNs

Los conceptos de FPNA yFPNN, pertenecen a los paradigmas orientados a simplificar las topologías de las redes neuronales con el fin de facilitar su implementación en hardware. Son modelos que aparecen como grafos de tareas parametrizadas, están especializados en realizar cálculos neuronales, pero difieren de los modelos neuronales estándar en que son grafos de regresiones no lineales. La distinción entre FPNA y FPNN está relacionada principalmente con el proceso de implementación: un FPNA es un conjunto dado de recursos neuronales que se organizan de acuerdo con relaciones específicas de vecindad, mientras que una FPNN es la forma de utilizar este conjunto de recursos modificando solo ciertos procesos de configuración local. Es decir, la implementación de dos FPNN en FPGA diferentes utilizan exactamente el mismo *mapeo* siempre y cuando estén basados en un mismo

5 Esta aritmética proporciona operadores diminutos, de modo que una red neuronal de tamaño medio puede ser implementada completamente en un FPGA, pero las restricciones de enrutamiento de actuales aún no lo permiten.

FPNA[60].

El primer objetivo del concepto FPNA es desarrollar estructuras neuronales que sean fáciles de mapear directamente en hardware digital gracias a una topología simplificada y flexible. La estructura de un FPNA está basada en los principios de los FPGA, es decir, funciones complejas realizadas mediante un conjunto de recursos programables simples. La naturaleza y las relaciones de estos recursos se derivan del procesamiento matemático que deben realizar los FPNA. Resumiendo el funcionamiento del modelo neuronal estándar visto en la §4.4.1, cada neurona calcula una función aplicada a una suma ponderada de sus entradas: Si $\vec{x}$ es el vector de entrada de la neurona i y $\vec{w}$ es el el vector de pesos, entonces calcula $f(\vec{w}_i, \vec{x}_i)$. El vector de entrada $\vec{x}$ puede contener tanto entradas de la red neuronal como salidas de otras neuronas, dependiendo del gráfico de conexión de la red. Por lo tanto, el número de entradas de cada neurona es también su cantidad de entradas (Fan-in) en el gráfico de conexión.

Por el contrario, los recursos neuronales en un FPNA se vuelven *autónomos*, lo que significa que sus dependencias se configuran libremente y el procesamiento resultante es más complejo que en los modelos neuronales estándar. Al igual que en los FPGA, la configuración de los FPNA maneja tanto las interconexiones entre recursos como la funcionalidad de estos.

5.5.1 *Recursos de los FPNA*

Un FPNA es un conjunto de recursos neuronales programables, que son definidos para el calculo de convoluciones parciales de regresiones no lineales. Dos tipos de *recursos* autónomos aparecen naturalmente en los FPNA: el primero son los **activadores** que aplican funciones neuronales estándar a un conjunto de valores de entrada y los segundos son los **enlaces** que se comportan como operadores independientes afines.

Estos dos *recursos* pueden manejarse de diferentes maneras, una forma sencilla sería asignar cualquier *enlace* a cualquier *activador*, con la ayuda de una red de interconección programable subyacente. No obstante esto conduciría a redes neuronales estándar de poda masiva, los problemas topológicos seguirían apareciendo (como los fan-in altos), y la distribución resultante conduciría a una poca diversidad de vectores de peso $\vec{w}_i$. Por lo tanto, los FPNA utilizan otro principio conocido por el nombre de **localidad**, en donde cualquier *enlace* puede estar conectado a cualquier *recurso local*. El objetivo de la **localidad** es reducir los problemas topológicos, y permitir que los enlaces conectados den como resultado vectores de peso más diversos. Más precisamente, los *enlaces* conectan los nodos de un grafo dirigido y cada nodo contiene un *activador*.

La particularidad de los FPNAs es que las relaciones entre los recursos locales de cada nodo se pueden establecer libremente; un enlace se puede conectar o no a un activador local o a otros enlaces locales. También, aparecen conexiones directas entre enlaces afines de modo que el FPNA calcula numerosas transformaciones

compuestas. Estas composiciones, crean numerosas conexiones neuronales virtuales de modo que se pueden obtener diferentes términos de convolución con un número reducido de pesos de conexión[60].

La siguiente definición especifica la estructura de un FPNA (como grafo dirigido), así como la naturaleza funcional de cada recurso neuronal individual.

Definición 5.5.1. Definición de FPNA:
Un FPNA se define por medio de[60]:

- Un grafo dirigido o digrafo, definido por un par de conjuntos $(\mathcal{N}, \varepsilon)$, donde $\mathcal{N}$ es un conjunto ordenado de nodos o vértices y ε es un conjunto de aristas dirigidas sin bucle. ε puede ser visto como un subconjunto de $\mathcal{N}^2$
- Para cada nodo n el conjunto de los predecesores directos de n se define como: $\mathrm{Pred}(n) = \{p \in \mathcal{N} \mid (p, n) \in \varepsilon\}$ y sus respectivos sucesores se definen como: $\mathrm{Suc}(n) = \{s \in \mathcal{N} \mid (n, s) \in \varepsilon\}$) El conjunto de nodos de entrada es: $\mathcal{N}_i = \{n \in \mathcal{N} \mid \mathrm{Pred}(n) = 0\}$.
- Un conjunto de operadores afines $\alpha_{(p,n)}$ para cada (p,n) en $\in$
- Un conjunto de activadores (i_n, f_n) para cada n en $\mathcal{N} - \mathcal{N}_i$, donde i_n es un operador de iteración (una función de $\mathbb{R}^2$ a $\mathbb{R}$) y f_n es una función de activación de ($\mathbb{R}$ a $\mathbb{R}$)

Por simplificación, (p, n) y (n) representan los enlaces y activadores correspondientes.

INTERPRETACIÓN: Los *recursos* (activadores y enlaces) están asociados con los nodos, mientras que la *localidad* está definida mediante los vértices. Para cada nodo $n \in \mathcal{N}$ existe un activador y tantos enlaces de comunicación como el número de predecesores de este nodo, y cada enlace esta asociado con un operador afín. Un activador está definido por (i_n, f_n) y manejará cualquier cálculo neuronal como en un programa secuencial; esto se basa en el hecho de que cualquier cálculo neuronal estándar puede realizarse mediante un bucle que actualiza una variable con respecto a las entradas de la neurona, así como un cálculo final que asigna esta variable a la salida de la neurona. La función de iteración (i_n) representa la función de actualización dentro del bucle. Finalmente, la salida de la neurona se calcula mediante la función (f_n).

A continuación se muestra un el ejemplo de un grafo dirigido sencillo (fig. 47) y como se definen sus componentes según la definición de FPNA :

- $\mathcal{N} = a, b, c, d, e, f, g$
- $\varepsilon = (a, b), (b, c), (b, e), (b, d), (d, f), (e, d), (g, e)$

Predecesores de e y d:

- $\mathrm{Pred}(n) = \{p \in \mathcal{N} \mid (p, n) \in \varepsilon\}$
  - $\mathrm{Pred}(e) = (g, e), (b, e)$

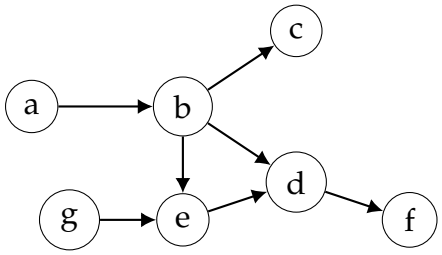


Figura 47: Ejemplo de grafo dirigido

- $\mathrm{Pred}(d) = (e, d), (b, d)$

Sucesores de b:

- $\mathrm{Suc}(n) = \{s \in \mathcal{N} \mid (n, s) \in \varepsilon\}$
  - $\mathrm{Suc}(b) = (b, c), (b, d), (b, e)$

Nodos de entrada $\mathcal{N}_i = a, g$

- $\mathcal{N}_i = \{n \in \mathcal{N} \mid \mathrm{Pred}(n) = 0\}$
  - $\mathrm{Pred}(a) = 0$
  - $\mathrm{Pred}(g) = 0$

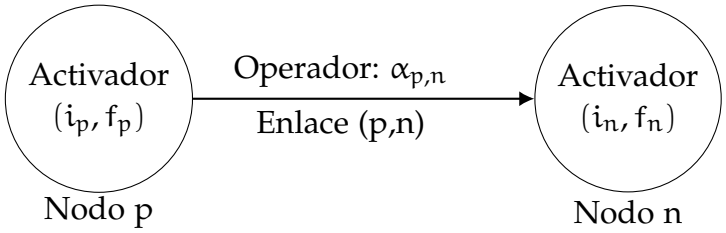


Figura 48: Enlace y activador para cada nodo

Cuando se han definido tanto el gráfico FPNA y los diferentes operadores se puede dar una implementación general, aquí cada recurso corresponde a un bloque básico y estos bloques se organizan de acuerdo con el gráfico; esta implementación puede ser utilizada por cualquier FPNN derivado de este FPNA. Algunos de los FPNA calculan funciones muy complejas (equivalentes a las redes neuronales estándar), sin embargo el gráfico FPNA es en sí una simplificación (número reducido de nodos, entradas y salidas por nodo limitadas) de modo que con el diseño del FPNA es posible mapear fácilmente la red neuronal en un dispositivo de hardware.

Lo más interesante del paradigma FPNA es que es un marco de cómputo neuronal adaptado al hardware y no es como otros métodos de implementación que manejan cálculos neuronales simplificados o usan formas astutas para contrarrestar los problemas de área y topología anteriormente mencionados. Los problemas de implementación no se resuelven con algunos límites impuestos a las arquitecturas neuronales o los cálculos neuronales: el esquema de computación FPNA original crea un puente entre las redes neuronales complejas y los dispositivos de hardware disponibles. Dado que este puente se basa en un paradigma de cálculo general, puede adaptarse a muchas opciones de implementación aritmética, de canalización,

etc.

En muchos casos, el diseño de un FPNA consiste primero en elegir una arquitectura neuronal estándar que se adapte a la aplicación, luego en determinar un FPNA que se ajuste al dispositivo de implementación y se pueda configurar para que sea funcionalmente equivalente a la red neuronal elegida.

Finalmente, el FPNA obtenido se asigna directamente al dispositivo de hardware gracias a una implementación completamente modular: se proporcionan bloques configurables predefinidos para cada recurso neuronal y se ensamblan de acuerdo con la arquitectura amigable con el hardware del FPNA. Este enfoque responde a los diversos desafíos de las implementaciones neuronales en FPGA (o más generalmente en hardware digital) de la siguiente manera:

TOPOLOGÍAS 2D FAN-IN: El esquema de cálculo FPNA contrarresta una topología neuronal simplificada mediante una mayor complejidad local de las interacciones entre los recursos neuronales. Esta complejidad puede ajustarse para que la topología simplificada se ajuste al hardware se debe encontrar un equilibrio, ya que el aprendizaje de FPNA se vuelve más difícil cuando aumenta la complejidad de las conexiones configuradas locales.

OPERADORES: Los FPNA utilizan los mismos operadores que las redes neuronales estándar, pero una topología simplificada está relacionada con una reducción de dichos operadores, de modo que los problemas de área e interconexión se resuelven simultáneamente.

ALMACENAMIENTO DE PESOS Y DESEQUILIBRIO DE TIEMPO/ÁREA: El paradigma de computación FPNA crea numerosas conexiones virtuales a pesar de un número reducido de enlaces. Por lo tanto, las simplificaciones de topología dan como resultado menos enlaces, es decir, menos pesos y multiplicadores. Además, cada peso se usa simultáneamente de diferentes maneras para crear las conexiones virtuales, de modo que los cálculos de FPNA den como resultado una parte reducida del tiempo de cálculo general para las multiplicaciones de peso de entrada.

5.5.2 *Recursos de los FPNNs*

Un FPNN es un FPNA configurado, un FPNA cuyos recursos se han conectado de una manera específica (se deben dar algunos parámetros para especificar el cálculo de cada recurso). En otras palabras, el FPNA subyacente a un FPNN es una arquitectura topológica completamente especificada de recursos neuronales, mientras que el FPNN es una forma determinada de especificar cómo interactuarán estos recursos para definir un comportamiento funcional[60].

Definición 5.5.2. Definición de FPNN:
La configuración de un FPNN requiere conexiones locales (de enlace a enlace, de enlace a activador o de activador a enlace), así como precisiones acerca del proceso iterativo realizado por cada activador (valor inicial, número de iteraciones). Por lo tanto, un FPNN se especifica mediante:

- Un FPNA (recursos neuronales disponibles)
- Para cada nodo n en $\mathcal{N} - \mathcal{N}_i$:
  - Un valor real θ_n (Valor inicial de la variable actualizada por la función de iteración i_n)
  - Un entero positivo a_n (número de iteraciones antes de que un activador aplique su función de activación)
  - Para cada p en $\mathrm{Pred}(n)$, dos valores reales $W_n(p)$ y $T_n(p)$ (coeficientes de operador afín $\alpha_{(p,n)}(x) = W_n(p)x + T_n(p)$)
  - $\forall\, p \in \mathrm{Pred}(n)$ un valor binario $r_n(p)$ (establecido en 1 si el enlace (p, n) y el activador (n) están conectados)
  - $\forall\, s \in \mathrm{Suc}(n)$ un valor binario $S_n(s)$ (establecido en 1 si el activador (n) y el enlace (n, s) están conectados)
  - $\forall\, p \in \mathrm{Pred}(n)\ \forall\, s \in \mathrm{Suc}(n)$, un valor binario $R_n(p, s)$ (se establece en 1 si los enlaces (p, n) y (n, s) están conectados),
- Para cada nodo de entrada n en N_i:
  - Un entero positivo c_n (número de entradas enviadas a este nodo)
  - $\forall\, s$ en $\mathrm{Suc}(n)$, un valor binario $S_n(s)$

Se han definido varios métodos de cálculo para los FPNN, los principios en común se pueden describir de la siguiente manera:

- Todos los recursos se comportan de manera independiente
- Un recurso recibe valores y para cada valor:
  - El recurso aplica su(s) operador(es) local(es),
  - El resultado se envía a todos los recursos vecinos a los que el recurso esté conectado localmente (un activador espera un valor antes de enviar cualquier resultado a sus vecinos).

Así mismo, las principales diferencias con el cálculo neuronal estándar son:

- Un recurso puede o no estar conectado a un recurso vecino. Estas conexiones locales son establecidas por los valores $r_n(p)$, $S_n(s)$ y $R_n(p, s)$
- Un enlace puede enviar valores directamente a otros enlaces de comunicación.
- Un recurso (incluso un enlace) puede manejar varios valores durante un solo proceso de cálculo del FPNN.

5.5.3 *FPNN de alimentación directa*

Un FPNN es de alimentación directa (feedforward) si las conexiones configuradas localmente entre sus recursos no contienen ninguna dependencia cíclica. Por lo tanto, un FPNN puede ser de alimentación directa incluso si su FPNA subyacente utiliza un grafo cíclico. Dos definiciones formales expresan lo anterior: La primera, define el grafo de dependencia de un FPNN, los nodos son los recursos neuronales (enlaces y activadores) y los vértices corresponden a las conexiones locales configuradas[60].

Definición 5.5.3. Sea $\phi = \{\mathcal{N}, \in, (\theta_n, i_n, f_n, W_n, T_n, a_n, R_n, r_n, S_n)_{n\in\mathcal{N}}\}$ un FPNN. su gráfico de dependencia $\mathcal{G}_D(\phi) = (\mathcal{N}', \in')$ está definido por:

- $\mathcal{N}' = \in \cup\{(n)|n \in \mathcal{N}\}$
- $\in' = \{((p,n),(n,s))|R_n(p,s) = 1\} \cup \{((p,n),(n))|r_n(p) = 1\} \cup \{((n),(n,s))|S_n(s) = 1\}$

De hecho, el esquema de cálculo de un FPNN sigue los vértices orientados de su gráfico de dependencia. Por lo tanto:

Definición 5.5.4. Sea ϕ un FPNN, es de alimentación directa si $\mathcal{G}_D(\phi)$ no incluye ningún ciclo.

El concepto de FPNA es un paradigma de cálculo neuronal adaptado al hardware, conduce principalmente a simplificaciones topológicas de arquitecturas neuronales estándar, por lo que es posible configurar varias redes neuronales diferentes a partir de un FPNA dado. Los recursos de un FPNA se definen para realizar los cálculos de las neuronas estándar sin embargo se comportan de manera autónoma, en consecuencia, se pueden crear numerosas conexiones sinápticas virtuales gracias a la aplicación de un protocolo de intercambio de datos de multidifusión a los recursos de una red neuronal dispersa. Este nuevo concepto de cálculo neuronal permite reemplazar una arquitectura neuronal estándar topológicamente compleja por una red simplificada.

En un FPNA aparecen dos tipos de recursos neuronales autónomos: las neuronas que aplican funciones neuronales estándar a un conjunto de valores de entrada, por un lado, y los enlaces de comunicación que se comportan como operadores afines independientes, por otro lado. En un modelo neural estándar, cada enlace de comunicación es una conexión entre la salida de una neurona y una entrada de otra neurona. El número de entradas de cada neurona es su entrada en el grafo de conexión. Por el contrario, los enlaces de comunicación y las neuronas se vuelven autónomos en un FPNA: sus dependencias son libremente programables.

Más precisamente, los enlaces de comunicación conectan los nodos de un gráfico dirigido, cada nodo contiene una neurona. La especificidad de los FPNA es que las relaciones entre cualquiera de los recursos locales de cada nodo pueden establecerse libremente.

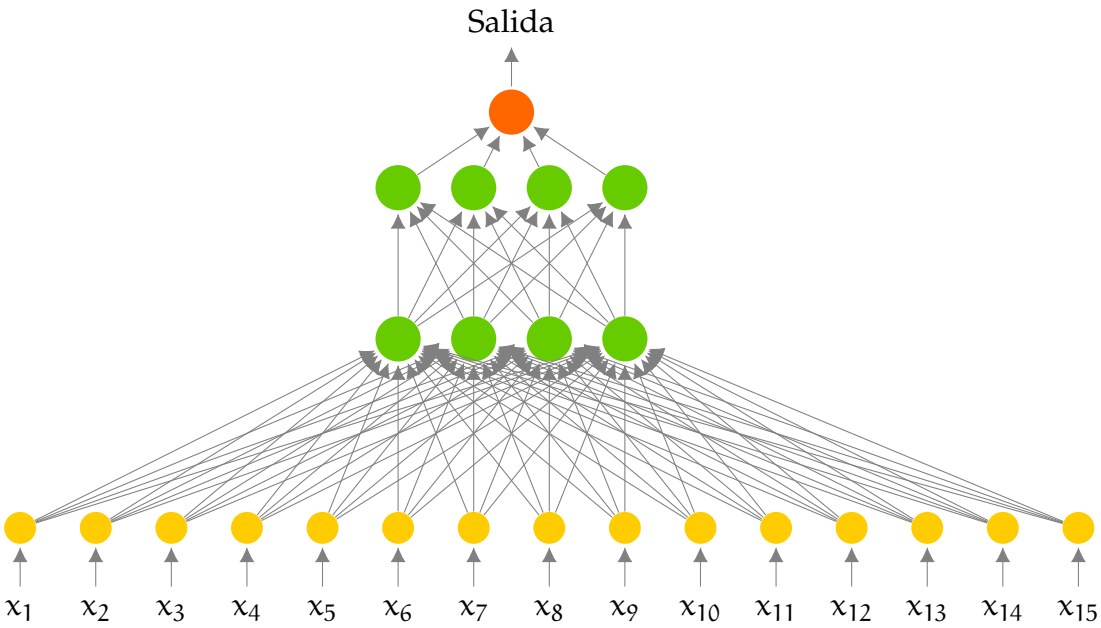


Figura 49: Perceptrón totalmente conectado para el problema de paridad (d=15)[60]

Una red neuronal puede verse como un grafo que contiene recursos neuronales interconectados (neuronas y pesos sinápticos). En un modelo estándar, el número de entradas de cada neurona corresponde al fan-in del gráfico de conexión. Por otro lado, los recursos neuronales (enlaces y activadores) se vuelven autónomos en un FPNA y sus dependencias se establecen de forma libre.

Los activadores aplican funciones neuronales estándar a un conjunto de valores de entrada y los enlaces se comportan como operadores afines independientes. El concepto FPNA permite conectar cualquier enlace a cualquier recurso local (cada nodo del grafo FPNA corresponde a un activador con enlaces entrantes y salientes). Aparecen conexiones directas entre enlaces afines, de modo que el FPNA calcula numerosas transformaciones compuestas afines, y a su vez, estas composiciones crean numerosas conexiones neuronales virtuales.

El activador de un nodo (n) realiza cualquier cálculo neuronal por medio de un bucle que actualiza una variable con respecto a las entradas de la neurona (i_n, denota la función de actualización) y un computo final (mediante la función f_n) que mapea esta variable a la salida de la neurona. Un enlace entre dos nodos p y n realiza una transformación afín $W_n(p)x + T_n(p)$.

5.6 Implementación y aplicación de FPNA's

Las matrices neuronales programables en campo se basan en un enfoque similar a FPGA: un conjunto de recursos cuyas interacciones se pueden configurar libremente. Estos recursos están definidos para realizar cálculos de neuronas estándar, pero se comportan de forma autónoma. Como consecuencia, se pueden lograr numerosos enlaces virtuales gracias a la aplicación de un protocolo de intercambio de datos de multidifusión a los recursos de una red neuronal dispersa. Este concepto de computación neuronal permite una arquitectura neuronal simplificada para re-

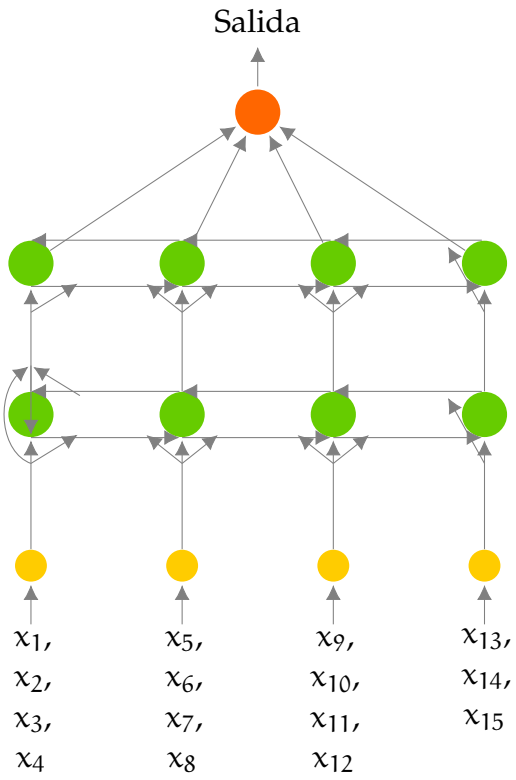


Figura 50: Arquitectura FPNN equivalente[60]

emplazar una virtual compleja.

En un FPNA aparecen dos tipos de recursos neuronales autónomos: neuronas que aplican funciones neuronales estándar a un conjunto de valores de entrada, por un lado, y enlaces de comunicación que se comportan como operadores afines independientes, por otro lado.

En un modelo neuronal estándar, cada enlace de comunicación es una conexión entre la salida de una neurona y la entrada de otra neurona, de modo que el número de entradas de cada neurona es su abanico en el gráfico de conexión. Por el contrario, los enlaces de comunicación y las neuronas se vuelven parcialmente independientes en un FPNA: sus dependencias son programables libremente.

A diferencia de los cálculos neuronales estándar, el paradigma FPNA permite que un recurso esté conectado o no a un recurso vecino. Además, un enlace de comunicación puede manejar varios valores y puede enviarlos directamente a otros enlaces.
Un FPNA es solo un conjunto no configurado de los recursos mencionados anteriormente. Por otro lado, un FPNN es un FPNA cuyos recursos se han conectado de una manera específica. Los principales parámetros de configuración son valores binarios: $r_n(p) = 1$ para conectar el enlace (p, n) y el activador (n), $S_n(s) = 1$ para conectar el activador (n) y el enlace (n, s), y $R_n(p, s) = 1$ para conectar los enlaces (p, n) y (n, s): Las conexiones directas entre enlaces crean enlaces virtuales compuestos.

Durante el cálculo del FPNN, todos los recursos (activadores y enlaces) se comportan de manera independiente. Cuando un recurso recibe un valor, aplica su(s) operador(es) local(es) y luego envía el resultado a todos los recursos vecinos a los que está conectado localmente. El poder computacional de los FPNN está vinculado a complejas relaciones de dependencia entre los pesos de las conexiones virtuales que se crean mediante conexiones locales directas entre enlaces. Se ha derivado un método de inicialización de peso dedicado para minimizar la influencia de estas dependencias complejas.

Una implementación FPNN solo requiere bloques predefinidos configurables que simplemente se ensamblan como en la arquitectura FPNA subyacente (mapeo paralelo directo de la arquitectura neuronal en el FPGA). Su topología simplificada hace posible que la herramienta de compilación funcione bien.

Se implementan los recursos (enlaces y activadores) del FPNA: cada recurso corresponde a un bloque predefinido que realiza cálculos de recursos y manejo de protocolo asíncrono, y todos los bloques se ensamblan de acuerdo con el grafo FPNA. Esta implementación puede ser utilizada por cualquier FPNN derivado de este FPNA: algunos elementos solo tienen que configurarse correctamente dentro de cada bloque, como multiplexores, registros, etc. Tales FPNN pueden calcular funciones complejas aunque su arquitectura FPNA es simple (número reducido de bordes, entradas y salidas limitadas).

Por lo tanto, este método de implementación es flexible, directo y compatible con las restricciones de hardware. Además, esta implementación muestra cómo el tiempo puede ser inherente a los cálculos de FPNN. La definición formal conjunta del método FPNA-FPNN seria la siguiente:

Definición 5.6.1. Definición de FPNA-FPNN:

- Un grafo dirigido o digrafo, definido por un par de conjuntos $(\mathcal{N}, \varepsilon)$, donde $\mathcal{N}$ es un conjunto ordenado de nodos o vértices y ε es un conjunto de aristas dirigidas sin bucle. Para cada nodo n el conjunto de los predecesores directos de n se define como: $\mathtt{Pred}(n) = \{p \in \mathcal{N} \mid (p, n) \in \varepsilon\}$ y sus respectivos sucesores se definen como: $\mathtt{Suc}(n) = \{s \in \mathcal{N} \mid (n, s) \in \varepsilon\}$) El conjunto de nodos de entrada es: $\mathcal{N}_i = \{n \in \mathcal{N} \mid \mathtt{Pred}(n) = 0\}$.
- Un grupo de neuronas $((\theta_n, i_n, f_n))_{n\in\mathcal{N}}$ donde $\theta_n \in \mathbb{R}$, i_n es una función de $\mathbb{R}^2$ a $\mathbb{R}$ y f_n es una función de $\mathbb{R}$ a $\mathbb{R}$: hay un recurso (por nodo) que maneja esencialmente cualquier calculo neuronal.
- Un conjunto de funciones afines $(x \mapsto W_n(p)x + T_n(p))_{(p,n)\in\varepsilon}$: para cada nodo hay tantos enlaces de comunicación como tantos predecesores tiene este nodo, cada enlace de comunicación está asociado con un operador afín.
- Para cada nodo de entrada n en $\mathcal{N} - \mathcal{N}_i$,
    - Un entero positivo a_n: número de iteraciones antes de que una neurona aplique su función de transferencia

  - $\forall\, p \in Pred(n)$ un valor binario $r_n(p)$ (establecido en 1 si el enlace (p, n) y el activador (n) están conectados)
  - $\forall\, s \in Suc(n)$ un valor binario $S_n(s)$ establecido en 1 si la neurona en (n) y el enlace (n, s) están conectados.
  - $\forall\, p \in Pred(n)\ \forall\, s \in Suc(n)$, un valor binario $R_n(p, s)$ se establece en 1 si los enlaces (p, n) y (n, s) están conectados

- Para cada nodo de entrada n en $\mathcal{N}_i$
  - Un entero positivo c_n: número de entradas globales enviadas por este nodo
  - Para cada s en $Suc(n)$, un valor binario $S_n(s)$

Las siguientes subsecciones describen la implementación del cálculo secuencial y paralelo asíncrono por medio de dos bloques predefinidos: recursos de enlace y activador.

5.6.1 *Cálculo secuencial*

Este método maneja una lista de tareas $\mathcal{L}$ que se procesan deacuerdo a una programación FIFO. Cada tarea $[(p, n), x]$ corresponde a un valor x enviado en un enlace (p, n).

El manejo de tareas corresponde a los diversos pasos que se deben realizar a nivel de nodo para lidiar con dicha entrada:

- Los nodos de entrada solo tienen que enviar valores de entrada globales a sus vecinos conectados.
- Un valor entrante se procesa primero de manera afín
- Luego se reenvía directamente a los nodos vecinos cuando existen conexiones directas entre enlaces.
- También es procesado por el activador local cuando existe una conexión entre el enlace entrante y este activador. Este último genera una salida cuando ha recibido suficientes entradas. Esta salida se envía a todos los vecinos conectados.

El algoritmo es el siguiente:

5.6.1.1 *Inicialización:*

ENTRADAS: Para cada nodo de entrada n en $\mathcal{N}_i$, los valores $(x_n^{(i)})_{i=1..c_n}$ de c_n son dados (entradas externas del FPNN) y las tareas correspondientes $[(n, s), x_n^{(i)}]$ se ponen en $\mathcal{L}$ para todos s en $Succ(n)$ tales que $S_n(s) = 1$. El orden de creación de la tarea corresponde a un orden lexicográfico (n, i, s) (según el orden en $\mathcal{N}$).

VARIABLES: Para cada nodo $n \in \mathcal{N} - \mathcal{N}_i$ se utilizan dos variables locales: c_n es el número de valores recibidos por el activador local, mientras que x_n es el valor que actualiza la función de iteración i_n. Inicialmente $c_n = 0$ y $x_n = \theta_n$

5.6.1.2 *Procesamiento secuencial*

Mientras $\mathcal{L}$ no esta vacía

Sea $[(p, n), x]$ el primer elemento en $\mathcal{L}$

1. Suprimir este elemento en $\mathcal{L}$
2. $x' = W_n(p)x + T_n(p)$
3. Para todo $s \in Succ(n)$ tal que $R_n(p, s) = 1$, cree $[(n, s), x']$ en $\mathcal{L}$ según el orden de sucesores s en $\mathcal{N}$
4. Si $r_n(p) = 1$
   - incremente c_n y actualize $x_n : x_n = i_n(x_n, x')$
   - Si $c_n = a_n$
     a) $y = f_n(x_n)$
     b) restablecer las variables locales $(c_n = 0, x_n = 0)$
     c) Para todo $s \in Succ(n)$ tal que $S_n(s) = 1$, cree $[(n, s), y]$ en $\mathcal{L}$ según el orden de los sucesores s en $\mathcal{N}$

Si $r_n(p) = 1$, se dice que el activador (n) recibe el valor de la tarea $[(p, n), x]$.

Un propósito a largo plazo del concepto FPNA es mostrar que se puede diseñar un chip neuronal reconfigurable y muy flexible, donde los CLBs se convierten en recursos neuronales.

Este documento propone un método de implementación basado en FPGA muy general que deriva directamente del esquema de cómputo paralelo asíncrono, para expresar con mayor claridad cómo el tiempo es inherente a los cálculos de FPNN.

Una implementación de FPNA puede ser modular, se proporcionan bloques predefinidos (recursos configurables), realizan cálculos de recursos y manejo de protocolo asíncrono, y se ensamblan de acuerdo con el gráfico FPNA. Dado que el concepto FPNA permite manejar aplicaciones complejas con topologías FPNA simples este método de implementación es flexible, directo y compatible con las restricciones de hardware.

Ejemplo: En la figura muestra una posible implementación de un enlace de comunicación; donde todos los flip-flops usan el reinicio asíncrono activo en 1.

SELECT : Este bloque recibe todas las señales de solicitud que pueden enviar los predecesores del enlace de comunicación de acuerdo con la topología FPNA. También recibe una señal libre, establecida en '1' cuando la comunicación no maneja ninguna solicitud. La señal `inicio` es una señal de salida establecida

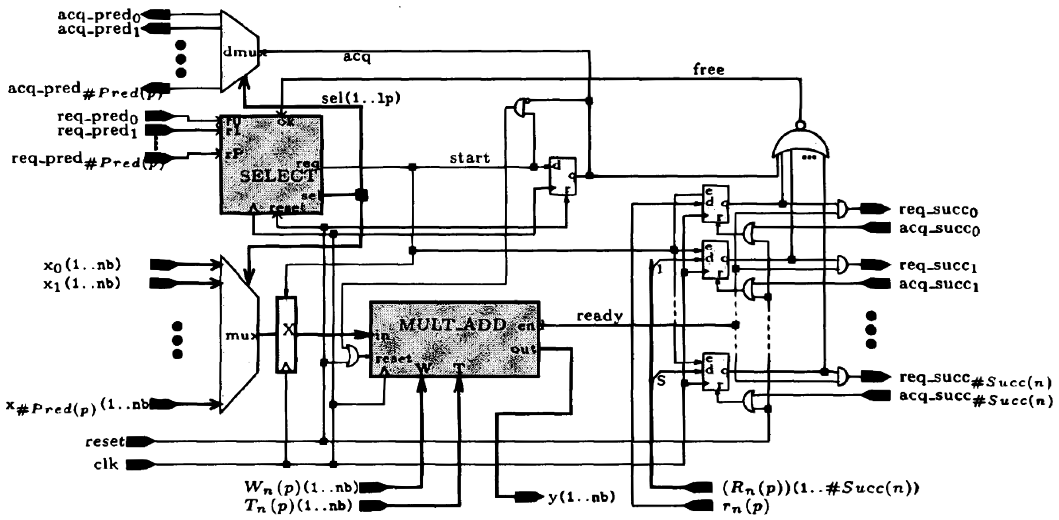


Figura 51: Arquitectura de un enlace de comunicación[60]

en '1' durante un ciclo de reloj cuando comienza el procesamiento de una solicitud. La señal `acq` es una señal de reconocimiento que se enruta hacia el recurso que envió la solicitud procesada actualmente. La señal `sel` (1.. Ip) codifica el número de la señal de solicitud seleccionada. Controla la entrada de `mux` y el reconocimiento de `dmux`.

MULT_ADD : Este bloque recibe el valor almacenado en el registro X (valor de la solicitud seleccionada). También recibe $W_n(p)$ y $T_n(p)$. Emite una señal `ready` que se establece en '1' cuando la señal y (1. .nb) contiene el valor de salida del enlace de comunicación (transformación afín de X).

FREE : Cuando el registro X almacena la entrada seleccionada, un conjunto de flip-flops almacena las señales de solicitud que deberán enviarse a los sucesores del enlace de comunicación. Estos flip-flops se reinician cuando se reciben los acuses de recibo correspondientes. La señal libre se restablece cuando se han recibido todos los reconocimientos esperados. Sin embargo, las señales de solicitud de salida efectiva permanecen bajas siempre que `ready` no esté activo.

El concepto FPNN-FPNA tiene como objetivo implementar un amplio espectro de arquitecturas de redes neuronales en el mismo hardware. El objetivo del concepto es diseñar una plataforma independiente del hardware objetivo.

Aunque actualmente el hardware digital programable presenta una oportunidad real para implementaciones flexibles de hardware de redes neuronales, aún surgen muchos problemas de área y topología cuando los modelos neuronales estándar se implementan en circuitos programables como FPGA, de modo que las rápidas mejoras de la tecnología FPGA no se pueden explotar por completo.

6

ESTADO DEL ARTE DE IMPLEMENTACIÓN DE REDES NEURONALES EN FPGA

En este capítulo se presenta el estado del arte sobre la implementación de redes neuronales en FPGA. El capítulo está dividido en dos partes principales, la primera resume una serie de metodologías utilizadas actualmente para solucionar los problemas de implementación haciendo énfasis en una serie de trabajos en los cuales se aplica la metodología FPNA-FPNN, y la segunda parte habla sobre las tecnologías FPGA de última generación que están siendo utilizadas actualmente en misiones espaciales.

6.1 Soluciones a problemas de implementación

Debido al gran número de neuronas que poseen las redes neuronales, su implementación aún sigue siendo una tarea difícil, comenzando por el hecho de que los algoritmos incluyen una gran cantidad de multiplicaciones. Como se vio en el capitulo 5, es común que las redes neuronales artificiales sean implementadas como software ejecutado en un procesador de señales o en una computadora de propósito general. Sin embargo, el migrar a un FPGA tiene el propósito de mejorar el desempeño del sistema aprovechando la naturaleza paralela de la red y del dispositivo, por lo que es no es raro encontrar diseños donde se aplican técnicas de computo paralelo, donde se describen múltiples procesadores y se utiliza el encauzamiento (pipeline), así como las técnicas de SIMD y MIMD. Por ejemplo, en [61] se compararan diversas implementaciones de ANN con aproximación sigmoidea, se proponen tres arquitecturas parcialmente paralelas y una completamente paralela, con ellas se realiza la propagación hacia atrás (BP) utilizando un FPGA *Virtex2000e* lenguaje Handel-C.

Sin embargo, utilizando estas técnicas de paralelismo no se resuelven necesariamente los problemas de implementación mencionados en la §5.4.1, relacionados con los recursos disponibles en un FPGA. No obstante, las implementaciones de ANN siempre buscan explotar, en lo posible, el hardware de los FPGAs. Recientemente, diferentes autores han implementado modelos en dispositivos FPGA con diferentes metodologías que dependen de la aplicación.

Una de las soluciones más sencillas para evitar problemas de implementación es considerar redes neuronales muy pequeñas, así como utilizar algunas de las siguientes soluciones estándar[85]:

- Implementar solamente de la fase de reconocimiento.

- Implementar un pequeño neuroprocesador en el FPGA y uso secuencial (y por lo tanto más lento) del dispositivo
- Implementar un pequeño neuroprocesador en cada uno de varios FPGAs con comunicación de datos, lo que aumenta los problemas de ancho de banda y crea la necesitad de minimizar la cantidad de E/S requeridas, lo cual es difícil cuando se manejan arquitecturas neuronales densamente conectadas.
- Utilizar cálculos simplificados, funciones de activación discretizadas y muy baja precisión.

Algunos ejemplos de estas soluciones pueden ser vistas en [73] donde se sugiere un método para diseñar e implementar un perceptrón multicapa (MLP) mediante unidades básicas de construcción basadas en una unidad multiplicador-acumulador (MAC), La red neuronal contiene tres capas con 240 neuronas en la capa de entrada; se implementa en un FPGA Xilinx Spartan-IIIE XC3S500E.

También, en [55] proponen un método preciso que utiliza un MPL simple (16 neuronas en la capa intermedia), para el reconocimiento de dígitos escritos a mano en idioma farsi, el método se implementa solo con operaciones de suma y resta, lo que acelera enormemente el proceso.

Por otro lado, en [73] se propone una codificación eficiente que conduce a la implementación de una arquitectura óptima de un MPL; utilizan la paralelización masiva, la codificación eficiente de la funciona sigmoide y el encauzamiento; la aplicación es clasificar, de forma multiespectral en un FPGA Virtex 6, imágenes satelitales adquiridas por el satélite argelino ALSAT-2A.

Por ultimo, en [50] Se desarrolló una arquitectura de hardware para redes neuronales pulsantes (SNN), el diseño contempla una arquitectura de hardware para emular una gran cantidad de neuronas, desarrollado módulos dedicados para la codificación, el aprendizaje y el reconocimiento en un solo chip.

No obstante, las soluciones antes descritas solo se adaptan a aplicaciones específicas con requisitos de precisión muy débiles. Muchas implementaciones neuronales en FPGA manejan cálculos neuronales simplificados y muchos métodos de implementación eficientes utilizados para implementaciones en ASIC o neurocomputadoras no pueden aplicarse libremente ya que estos tienden a limitarse a unas pocas arquitecturas neuronales bien adaptadas.

Dado que los principales problemas de implementación están vinculados a operadores codiciosos de área y topologías complejas, aparecen dos tipos de paradigmas de computación neuronal mutuamente complementarios: por un lado los marcos de cómputo neuronal que ahorran área, y por otro, los paradigmas que pueden manejar topologías simplificadas de la red.

6.1.1 *Soluciones de área*

Entre las soluciones para economizar el área disponible en los dispositivos FPGA está la optimización de la función de activación, ya sea a través del algoritmo de CORDIC[2], mediante la aproximación polinómica por partes controlada a través de tablas de búsqueda[23] o utilizando multiplicadores rápidos como se aprecia en [70]. La elección del método depende de las características de la aplicación así como los requerimientos, y pueden combinarse con otros métodos mas avanzados a su vez[60].

La elección de una aritmética adecuada puede verse reflejada en un mejor aprovechamiento de los recursos; se ha utilizado la aritmética en serie de bits tanto estándar[16] como optimizada[76], y la aritmética en serie de línea[23]. Debe tomarse en cuenta que cualquier aritmética en serie requiere más ciclos de reloj y debe usarse con operadores encausados[60].

También se han propuesto soluciones basadas en frecuencia en [28, 30] y aritmética basada en pulsos en [4, 13, 34, 49, 67, 69]. Esta clase de métodos permiten proporcionar pequeños operadores con los cuales es posible implementar redes neuronales de tamaño medio en un solo FPGA. Sin embargo, todavía resulta difícil implementar redes grande debido a las restricciones de enrutamiento[60].

6.1.2 *Soluciones de topología*

Se han realizado trabajos con el fin de encontrar soluciones topológicas que sean amigables con el hardware; entre ellas se encuentra la partición del grafo neuronal en sub-grafos equivalentes, de modo que mediante el uso repetido de una subgrafo simple se cubra al final todo el grafo en su totalidad[24].

Otro paradigma de calculo neuronal corresponde al concepto FPNA-FPNN descrito en la §5.5, el cual permite simplificar las arquitecturas de los modelos neuronales estándar sin una pérdida significativa de su capacidad de aproximación[22]. Tal es el caso de [21] donde se elige un problema simple de clasificación de patrones para mostrar cómo el método FPNA permite reemplazar arquitecturas neuronales estándar complejas por estructuras neuronales amigables con el hardware. Así mismo en [19] donde se describe un circuito analógico a una red neuronal original mediante el método FPNA, con el cual se puede emular con precisión grandes celdas complejas, o múltiples menos complejas.

El articulo [9] centra en un circuito novedoso adecuado para la implementación de redes neuronales artificiales de alimentación directa utilizando el concepto FPNA; describe un enfoque innovador para implementaciones rápidas de hardware de ANN, Introduce algunas técnicas especiales utilizadas en el diseño, así como su optimización. El circuito se optimiza para implementarse en FPGA modernos de Xilinx, principalmente las familias 5 y 6. También, en [35] se presenta un conjunto de métodos para *mapear* las redes neuronales entrenadas en FPNNs; estos métodos utilizan información obtenida de las redes originales, tales como la estructura de la

red, los pesos sinápticos y los sesgos.

Incluso se pueden encontrar diseños IP para la implementación de ANN en FPGA basados en el método FPNA, tal es el caso de [25] donde con en el diseño del núcleo IP de la ANN reutilizable, no solo se pueden ahorrar muchos recursos de hardware y mejorar la velocidad de procesamiento, sino que también el ciclo de desarrollo es rápido y posee una buena reconfigurabilidad, además de otras ventajas.
Los esfuerzos realizados para resolver los problemas en cuanto a las restricciones de implementación de ANN, abre un área de investigación futura, tal como se explica en [85] se deben crear puntos de referencia para poder comparar y analizar el desempeño de diferentes implementaciones, así como herramientas de software, que son necesarias para facilitar el intercambio de bloques de IP y bibliotecas de implementacion para ANN en FPGA; es necesario encontrar algoritmos de aprendizaje amigables con los dispositivos FPGA, y enfoques de adaptación topológica que contengan multiplicadores especializados y unidades MAC, para así aprovechar las características de los FPGA de ultima generació, Así mismo se puede lograr una mejora de la densidad, la cual se logra con métodos que aumenten la cantidad de funcionalidad efectiva de un circuito por unidad de área, todo esto a través de la reconfiguración del FPGA, lo anterior se logra explotando el tiempo de ejecución de un FPGA mediante cualquiera de los siguientes métodos:

- Multiplexar en el tiempo un chip FPGA para cada uno de los pasos secuenciales en un algoritmo ANN.
- Multiplexar en el tiempo un chip FPGA para cada uno de los circuitos de la ANN que estén especializados con diferentes operandos en determinadas etapas de la ejecución

Seleccionar la precisión de los pesos sinápticos también es una de las cuestiones importantes al implementar ANN en FPGA. Una mayor precisión en los pesos significa menos errores de cuantificación en las implementaciones finales, mientras que una menor precisión conduce a diseños más simples, mayor velocidad y reducciones en los requisitos de área y consumo de energía.

Por otro lado, la implementación de la función de transferencia de forma directa, en especial para las funciones de transferencia sigmoideas, es muy costosa, las funciones sigmoideas se pueden realizar mediante una serie de operaciones de desplazamiento y suma; muchas implementaciones de funciones de transferencia se realizan mediante aproximaciones lineales por partes. Otro método, son las tablas de búsqueda, en las que las muestras uniformes tomadas del centro de una función sigmoidea se pueden almacenar en una tabla para posteriormente buscarlas[60].

6.2 Tecnologías

Seleccionar los componentes correctos para cualquier proyecto de ingeniería es una elección crítica y difícil, más aún cuando se deben seleccionar componentes que se utilizarán en proyectos de vuelo de alto riesgo como las misiones espaciales.

Las necesidades de los ingenieros de diseño del JPL fueron la principal fuerza impulsora detrás del cambio de paradigma tecnológico de ASIC a FPGA. Los ingenieros del JPL Han tenido que seleccionar componentes utilizados en proyectos de vuelo de alto riesgo, y en misiones de alto perfil como la Misión Mars Exploration Rover. Los FPGA de Xilinx han sido diseñados y volados, desde hace tiempo, tal como los módulos de aterrizaje y los rovers de la NASA[64].

Los FPGA de Xilinx brindan ventajas de diseño inherentes para satisfacer los principales requerimientos de las misiones espaciales: alta densidad de compuertas lógicas, resistencia a la radiación, múltiples estándares de E/S y gran disponibilidad; lo que ha impulsado la transición tecnológica a medida que estos dispositivos pasan más y mas pruebas de verificación.

Los resultados obtenidos se se pueden ver reflejados en *Mars Exploration Rovers*, dentro de cada vehículo explorador (Discovery y Spirit), se encuentran dos Virtex XQVR1000 que sirvieron como los cerebros principales y para el control de los motores; cuatro dispositivos Xilinx XQR4062XL de la familia 4000XL controlaron la pirotecnia del módulo de aterrizaje en Marte, un punto crucial para el exitoso procedimiento de descenso y aterrizaje. También es evidente, aunque no tan visible, el creciente número de futuras misiones de vuelo que los ingenieros de JPL están diseñando con FPGA tolerantes a la radiación Xilinx[64].

En la actualidad se está utilizando software autónomo. Ejemplos de El software autónomo presente en el rover Mars Science Laboratory (MSL) es un software AEGIS (Exploración Autónoma para Recopilar Ciencia Incrementada), que ayuda al rover a elegir de forma autónoma objetivos para el láser[18] como se muestra en la Figura 1. Se pretende que haya otras formas de software autónomo para uso en visión, particularmente detección de partículas y para localización[3, 48].

JPL está trabajando actualmente en diseños de vuelo tanto con Virtex-II , la última familia tolerante a la radiación Xilinx, como con FPGAs tolerantes a la radiación Virtex. Estas nuevas misiones volarán en los próximos dos a cinco años y se están volviendo cada vez más sofisticadas tanto en los requisitos electrónicos de la misión como en la implementación del diseño.

Las nuevas misiones tienen requisitos más exigentes: más velocidad y más integración, con un objetivo continuo de menos espacio y menos peso. En el frente de Xilinx, cada familia posterior de FPGA tolerantes a la radiación (como los dispositivos Virtex-II Pro) proporcionará más integración, más características arquitectónicas y más capacidades.

6.3 Conclusión

El paralelismo mostrado por los modelos ANN se pierde cuando estos modelos se implementan en computadoras con esquemas de procesamiento secuencial. Cuando se implementan en hardware, las redes neuronales pueden aprovechar al máximo su paralelismo inherente y ejecutar varios órdenes de magnitud más rápido

que la simulación de software, volviéndose adecuados para aplicaciones del mundo real. Una implementación basada en FPGA de ANN con gran número de neuronas sigue siendo una tarea difícil porque los algoritmos de ANN son ricos en multiplicación y es relativamente costoso implementar multiplicadores en FPGA de grano fino.

En la literatura se encuentran varias implementaciones de hardware para ANN, pero las dos más relevantes según la implementación de hardware que se propone en este trabajo son: [21-23], donde se informa de la utilización de la metodología FPNA/FPNN con el cual es posible mapear de manera *amigable* redes neuronales en dispositivos FPGA.

Parte II

DISEÑO PRELIMINAR

Cuando estoy trabajando en un problema, nunca pienso en la belleza, pero cuando termino, si la solución no es hermosa, sé que está mal.

— R. Buckminster Fuller

7

METODOLOGÍA DE DISEÑO

En este capítulo, se presenta la metodología utilizada en el proceso de diseño del sistema técnico, se explica detalladamente cada uno de los pasos que se siguieron, así como su relación con las partes que conforman el documento de la tesis. También, se hace hincapié en las iteraciones realizadas durante el diseño y se finaliza incluyendo una explicación del nicho tecnológico al que pertenece el sistema desarrollado. Así mismo, en al final de este capítulo se incluye el diseño de concepto para dar inicio así a la etapa de diseño preliminar.

El diseño es un proceso iterativo y de constante toma de decisiones; algunas veces estas deben tomarse con muy poca información disponible y en ocasiones con un exceso de información parcialmente contradictoria. El diseño en ingeniería se define como el proceso de aplicación de varias técnicas y principios científicos con la finalidad de definir un dispositivo, un proceso o un sistema con el detalle suficiente para lograr su realización[**ulrich2013**]. El producto a diseñar se le conoce por lo general con el nombre de *sistema técnico*.

Definición 7.0.1. Sistema técnico Es el mecanismo complejo de acciones orientadas intencionalmente a la transformación de objetos concretos, para conseguir un resultado determinado. También se puede decir que el sistema técnico es el *proceso* de elaboración de un producto.

NOMENCLATURA DE LOS SISTEMAS: Con el fin de tener una mayor claridad y control durante el proceso, se decidió establecer un sistema de nomenclatura para el sistema y sus divisiones (ver anexo C). Para ello se adoptó una de las reglas de la NASA sigue para bautizar a sus proyectos:

«El nombre de cada proyecto será una simple palabra eufónica que no se duplicará ni se confundirá con otros títulos de proyectos. Cuando sea posible y si corresponde, se elegirán nombres para reflejar la misión» [**NASA**].

Así bien, para fines de identificación de este trabajo, el sistema técnico desarrollado fue designado con el nombrado: **Orfeo**, que de acuerdo al anexo C, se clasifica como un ensamble perteneciente al subsistema de visión artificial del rover.

7.1 División de la metodología de diseño

El procesos de diseño está estructurado en una serie de pasos, con el fin de establecer una metodología la cual se siguió durante todo el trabajo. Está compuesto por diez pasos primordiales que se enumeran a continuación:

1. Definición del problema

2. Identificación de la necesidad
3. investigación preliminar
4. Requerimientos y restricciones de diseño
5. Selección de la solución
6. Desarrollo de Concepto
7. Diseño a nivel de sistema
8. Diseño de detalle
9. Pruebas y verificación
10. Reporte de resultados

El proceso da inicio con los pasos 1 y 2 en donde se estableció el problema a resolver y se identificaron las necesidades generales y particulares. A continuación en el paso 3, se realizó una investigación preliminar con el fin de obtener información requerida que define y permite comprender por completo el problema a resolver; este tercer paso es cubierto por el marco teórico y el estado del arte.

El paso 4 consistió en recopilar un conjunto detallado de requerimientos y especificaciones de funcionamiento, las cuales delimitaron el problema y determinaron su alcance; estas especificaciones están divididas en dos tipos: *especificaciones objetivo* y *especificaciones finales*. Durante el paso 5, selección de la solución, se sintetizaron y analizaron todos los enfoques de diseño alternativos para solucionar el problema, y así seleccionar una solución viable. Así mismo, el desarrollo del concepto se completó en el paso 6 con su receptiva evaluación.

Posteriormente, en el paso 7, se realizó el diseño a nivel de sistema, donde el sistema técnico fue analizado mediante bloques funcionales. Posteriormente, cada una de las partes que conforman el sistema fueron desarrolladas en detalle en el paso 8; en este paso también se ataron algunos cabos sueltos presentes en el desarrollo de concepto. El proceso de diseño termina con las pruebas y verificación mediante simulaciones en el paso 9, a través de diversos experimentos. Por último, el paso 10, se refiere al reporte de resultados y conclusiones, así como la elaboración de la documentación que conforma en si la tesis.

El proceso de diseño descrito anteriormente no es lineal, por el contrario, se necesitó realizar varias iteraciones en las cuales se regresó desde cualquier paso a otro anterior. Los diez pasos descritos arriba cuentan con subpasos o pasos intermedios que amplían así el proceso de diseño con tareas sobresaliente desarrollas en cada uno de los pasos principales; dando así origen a veinte fases las cuales son agrupadas a un nivel superior en cuatro etapas. En la tabla 13 se presenta en forma clara las divisiones del proceso de diseño completo en este trabajo.

Tabla 13: División del proceso de diseño

Número	Fases	Etapas
1	Definición del problema	
2	Identificación de la necesidad	Etapa de definición
3	Investigación Preliminar	
4	Planteamiento de la meta	
5	Requerimientos y restricciones	
6	Desarrollo de concepto	
7	Análisis y Síntesis	Etapa de diseño preliminar
8	Diseño a nivel sistema	
9	Especificaciones	
10	Modelos Matemáticos	
11	Diagramas de bloques	
12	Programación	
13	Simulaciones	Etapa de diseño detallado
14	Experimentos	
15	Análisis y refinamiento	
16	Pruebas	
17	Verificación	
18	Elaboración de diagramas	
19	Trabajo escrito	Etapa de documentación
20	Presentación y resultados	

7.1.1 *Etapas del diseño*

Las grandes etapas que conforman la metodología de diseño también conforman el esqueleto de esta tesis, la cual también esta dividida en cuatro grandes partes (I, II, III y IV). A continuación, se explica a detalle como es que las grandes etapas de diseño están relacionadas con las cuatro partes de este documento:

ETAPA DE DEFINICIÓN: Esta etapa es cubierta en su totalidad por la parte I (*antecedentes*) conformada por los seis primeros capítulos de la tesis, y la primera mitad del capitulo 8 (requerimientos).

ETAPA DE DISEÑO PRELIMINAR: Es desarrollada en la parte II que lleva el mismo nombre; esta conformada por la sección 7.2 (desarrollo de concepto) y los capítulos 8 y 9, especificaciones y diseño a nivel de sistema, respectivamente.

ETAPA DE DISEÑO DETALLADO: En esta etapa se desarrollaron a profundidad cada una de las partes que conforman el sistema, es cubierta por los capítulos 10 y 11. Donde se creó un capitulo para cada una de las plataformas que integran el sistema técnico Orfeo.

ETAPA DE DOCUMENTACIÓN: Consiste en la escritura de la tesis en si, así como los capítulos 12 13, 14, donde se presentan los resultados, se dan con conclusiones y se plantea el trabajo futuro. También, incluye la parte de anexos y apéndices que contiene diagramas y hojas de datos.

7.2 Desarrollo de concepto

El concepto del sistema partió del planteamiento de la meta la cual es: diseñar un sistema de reconocimiento de patrones mediante la implementación de hardware de una red neuronal, con la finalidad de identificar terrenos encontrados en el ambiente marciano, y así contribuir a la tarea de navegación autónoma de los rover exploradores de Marte.

El sistema consiste en una red neuronal[1] implementada en FPGA, la cual es entrenada fuera de chip en una computadora de propósito general utilizando lenguaje de alto nivel. El entrenamiento radica en presentar una serie de imágenes, previamente etiquetas del terreno marciano; tomadas del las bases de imágenes de misiones pasadas de la NASA. Paralelamente se tiene el diseño en hardware de la red neuronal, al cual se le exportan los parámetros óptimos obtenidos en el entrenamiento, así como los valores de los pesos sinápticos.

La comprobación del sistema consiste en dos tareas principales: la primera, es probar el sistema de visión artificial utilizando imágenes reales del terreno marciano, determinar su desempeño y porcentaje de error de reconocimiento. La segunda, probar el funcionamiento de cada uno de los módulos diseñados verificando que se cumplan los requerimientos establecidos.

Al reconocer las fotografiaras del terreno, la red neuronal examina intrínsecamente ciertos características propias de cada terreno, por ejemplo tamaño de grano, textura; de esta manera reconoce lo que esta viendo y le da información al sistema de navegación, el cual tomará las decisiones pertinentes para para adaptarse mejor al tipo de terreno al que se enfrenta o tomar acciones si se encuentra con terrenos que representen un peligro para la integridad del robot.

1 En este trabajo se propone utilizar el clasificador neuronal LIRA para reconocimiento de texturas del terreno marciano.

8

REQUERIMIENTOS

En este capitulo se presentan los requerimientos y restricciones para el diseño del sistema técnico, estas fueron establecidas a partir de una lista general de necesidades desarrollada a partir de la declaración de la misión, presentada aquí mismo. También, se tomaron como base la documentación de sistemas probados en misiones exitosas de la NASA para definir los requerimientos y especificaciones de diseño para sistemas espaciales. El capítulo termina con la tabla de especificaciones objetivos para diseñar el sistema técnico.

Como primer paso antes de establecer los requerimientos y especificaciones de diseño, se especificó una nicho de oportunidad particular; la información se encuentra formalizada como la *declaración de la misión* (a veces también llamada carta o reporte de diseño).

La declaración de la misión sirve para indicarnos en qué dirección ir; sin embargo, no dice una forma particular de avanzar. La definición de la declaración de la misión es el resultado de las actividades de investigación y planeación descritas en los capítulos anteriores; la declaración de la misión del proyecto se ilustra en la tabla 14:

Tabla 14: Declaración de la misión

	Sistema técnico: Orfeo
Descripción del producto	·Ensamble para el reconocimiento de terreno
Propuesta de valor	·Entrenable, alta velocidad de op., impl. de bajo costo
Metas clave	·Probar el sistema con imágenes reales de Marte ·Implementación en hardware · Lograr un porcentaje de reconocimiento $>$ al 95 %

8.1 Necesidades

El proceso de identificar las necesidades es parte integral del proceso de desarrollo del producto, está estrechamente relacionado con la generación de conceptos, la selección del concepto, y el establecimiento de especificaciones del producto. Las necesidades identificadas principalmente se muestran a continuación en forma de lista. Estas tienen relación con otras actividades iniciales tales como el desarrollo del concepto.

8.1.1 *Lista de Necesidades*

- Capacidad de adquirir y procesar imágenes
- Reconocimiento de texturas mediante métodos de machine learning
- Capaz de almacenar las imágenes a procesar.
- El sistema debe operar con bajo consumo de energía
- Ligero y compacto
- Trabajar con imágenes reales tomadas de la superficie de Marte
- Debe operar velozmente
- Que integre tecnologías equivalentes utilizadas en misiones espaciales reales
- Capacidad de implementar diferentes algoritmos sin cambiar el hardware utilizado
- Que pueda actualizarse
- Escalable a sistemas de mayor complejidad
- Que cuente con un procedimiento sistematizado para su configuración
- Alto porcentaje de reconocimiento

Un sentido de la importancia relativa de las diversas necesidades enlistadas anteriormente, es esencial para realizar las concesiones de manera correcta. Por lo tanto, el siguiente paso en el proceso fue establecer la importancia relativa de las necesidades identificadas; el resultado de este paso es una valoración numérica de importancia (ver tabla 15), la evaluación de importancia fue realizada en base a las tareas y objetivos principales del sistema así como del conocimiento que se tiene sobre la de operación esperada del prototipo final.

Tabla 15: Necesidades

Número	Parte	Necesidad	Imp.
1	Electrónica	Equiv. tecnológica con misiones anteriores	5
2	Electrónica	Bajo consumo de energía	5
3	Electrónica	Escalable	5
4	Sistema	Reconfigurable	5
5	Sistema	Flexible	4
6	Sistema	Diseño modular	5
7	Sistema	Robusto	4
8	Sistema	Configuración sistematizada	4
9	Plataforma	Ligera y compacta	5
10	Datos	Adecuado para trabajar en tiempo real	5

11	Datos	Procesamiento veloz	5
12	Datos	Almacenamiento provisional	5
13	Datos	Utilizar la base de imágenes PDA Y PIA	5
14	Datos	Porcentaje de reconocimiento	5

8.2 Requerimientos y restricciones

Los requerimientos de diseño para las cámaras de ingeniería de los rover se derivan de una serie de fuentes, incluida la experiencia operativa obtenida durante la misión Mars Pathfinder, y los estudios de diseño del proyecto MER (Mars Exploration Rover), se establece así algunos requisitos generales para proyectos futuros tales como la distancia transversal del rover o la precisión de navegación.

Tabla 16: Requerimientos Generales

Código	Descripción
ORF.G.01	Procesamiento de imágenes de la base de datos PDS
ORF.G.02	Rango de temperatura
ORF.G.03	Actualizable
ORF.G.04	Reconfigurable
ORF.G.05	Escalable

Tabla 17: Requerimientos funcionales

Código	Descripción
ORF.F.01	Consumo de potencia menor a 1 watt
ORF.F.02	Capacidad de memoria mínima para almacenar dos imágenes por cada clase a reconocer
ORF.F.03	Procesamiento de imágenes en escala de grises

Tabla 18: Requerimientos de desempeño

Código	Descripción
ORF.D.01	Contar con la mayor velocidad de transmisión posibles de datos hacia el sistema
ORF.D.02	Que el reconocimiento de cada imagen tarde menos de un segundo
ORF.D.03	El porcentaje de reconocimiento de cada escena debe superar el 90 %

Así mismo, se creó un sistema de código para identificar cada uno de los requerimientos y tener un mejor control en cuanto al cumplimiento de estos. EL código se lee de la siguiente manera: Las primera parte son las tres primeras letras en mayúscula del nombre del sistema técnico seguidas de una letra (G,F o D) que designan si el requerimiento es general, funcional o de desempeño respectivamente y la última parte del código es el número de requerimiento; cada una de las partes es separada por un punto.

8.3 Especificaciones

Las necesidades se expresan generalmente en lenguaje coloquial; no obstante, mientras que esas expresiones son útiles para crear un sentido claro de los problemas que son de interés, sirven de muy poco respecto a cómo diseñar y construir el producto. Por esta razón, fue necesario establecer un conjunto de especificaciones que expliquen con detalles precisos y mesurables lo que el producto tiene que hacer; las dos cualidades principales cada especificación son su métrica y su valor.

8.3.1 *Especificaciones Objetivo*

La primeras especificaciones que se definieron fueron las especificaciones objetivo o *especificaciones meta*. Estas especificaciones se establecieron antes de conocer cuales son las restricciones que impone la tecnología para desarrollar el sistema; las especificaciones objetivo están basadas en valores de sistemas similares encontrados en el estado del arte y en la investigación previa realizada en el marco teórico. En la tabla 19 se presentan las especificaciones objetivo de cada una de las partes del sistema diseñado. En la tabla, se manejan dos tipos de valores para las métricas: un *valor ideal* y un *valor marginalmente aceptable*. El primero es el mejor resultado que se puede esperar, mientras que el segundo es el valor que apenas haría aceptable el sistema en termino de competencia con los ya existentes.

Tabla 19: Especificaciones objetivo del sistema

Núm	Núm. de Necesidad	Métrica	Imp.	Unid.	Valor Marginal	Valor Ideal
1	13	Tamaño de la imagen	5	píxel	-	128×128
2	1,13	Rango espectral	4	nm	+/- 50	600-800
3	12,14	Capacidad de memoria	5	Mbyte	>10	>15
4	10,11	Velocidad de reconocimiento	5	s	<1.5	<1
5	14	Porcentaje de reconocimiento	5	%	>88	>95
6	2	Potencia	4	W	<1.5	<1
7	4,7	Capas	5	Num	-	4
8	4,7	Dimensiones de capa A	4	Neuronas	32000	64000
9	13	Imágenes para entrenar	5	Num	25	30
10	13	Imágenes para reconocimiento	5	Num	5	10

8.3.2 *Especificaciones Finales*

Las especificaciones que anteriormente fueron expresadas en la tabla 19 como amplios rangos de valores, son refinadas y conformadas de manera más precisa como especificaciones finales. Estas especificaciones se establecieron después de la selección de concepto y el diseño preliminar. Así como de los de los modelos técnicos y las primeras pruebas.

9 DISEÑO A NIVEL DE SISTEMA

En la vida real los sistemas existen como parte de uno o varios sistemas más grandes; así mismo, es necesario dividir estos subsistemas en componentes o partes más pequeñas a su vez con el fin de facilitar el proceso de desarrollo. En este capítulo, se especifica el conjunto de componentes que interactúan de manera sinérgica y conforman el sistema técnico Orfeo. Así también, se generar la arquitectura del producto, se definen los subsistemas e interfaces. Por último se realiza un análisis de fabricar contra comprar, se identifican los proveedores para componentes y se define el esquema final de ensamble.

9.1 Estructura del sistema

Se siguió una metodología de diseño Top-Down, donde se captura la idea en un alto nivel de abstracción, posteriormente se especificaron los bloques funcionales y las interconexiones que la conforman, y finalmente fueron diseñados cada uno de los componentes incrementando cada vez más el nivel de detalle. Las ventajas que se tienen al elegir este método de diseño son que: incrementa la productividad del diseño, incrementa la reutilización del diseño, y se logra una rápida detección de errores.
A partir de la tabla de necesidades, requerimientos y especificaciones objetivo presentadas en el capitulo 8, así como la declaración de la misión (sección 14) y los objetivos establecidos para este trabajo; se genero una estructura de bloques con las principales partes que debe contener el sistema para realizar la tarea planteada.

La tarea principal del sistema es reconocer patrones de los tipos de terrenos encontrados en el planeta Marte, debido a que no es posible adquirir las imágenes directamente para este trabajo, el sistema utiliza como datos de entrada las imágenes pertenecientes a las bases de datos de la NASA (ver sección 1.4). Por lo tanto, el diagrama de la figura 52 donde se parte de un objeto en la escena se simplifica tal como se muestra en la figura 53.

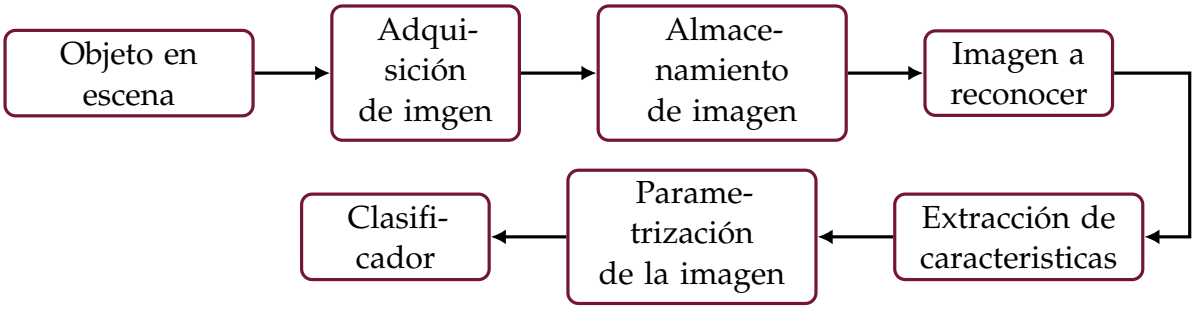


Figura 52: Diagrama de bloques del sistema, adquisición y reconocimiento

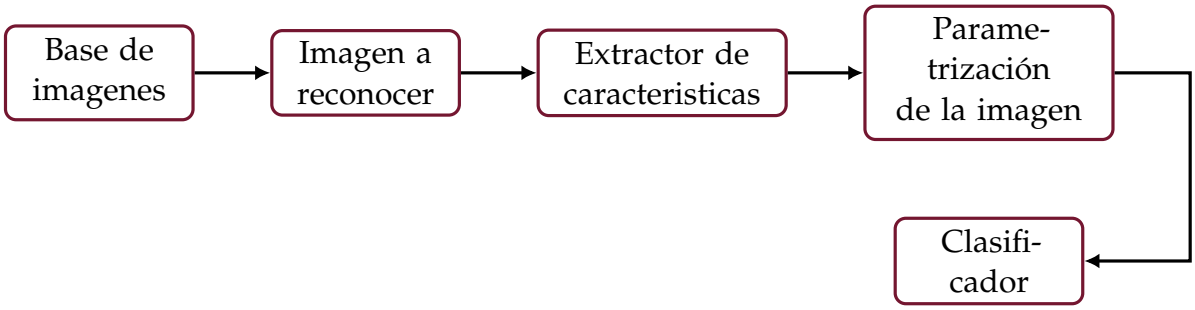


Figura 53: Diagrama de bloques del sistema simplificado

El **clasificador** es la parte más importante del sistema, puede decirse que es el pivote a partir de donde el sistema comienza a diseñarse; si tomamos como base las necesidades relacionadas con tener un sistema que trabaje de forma autónoma, veloz, y que cuente con las características de ser actualizable y escalable, entonces, esto se traduce a utilizar métodos de *aprendizaje automático.*

Los algoritmos de redes neuronales artificiales (ANN), han demostrado tener muy buenos resultados para tareas de reconocimiento de patrones en el pasado; por esto mismo, se decidió que el sistema de reconocimiento tenga como base un algoritmo de este tipo.

La red neuronal propuesta para el trabajo es la red conocida con el nombre de **C**lasificador **N**euronal de **A**rea **R**eceptiva **L**imitada (LIRA, por sus siglas en inglés), específicamente su versión en escala de grises (LIRA grey-scale), es una red neuronal artificial basada en el perceptrón de Rosenblatt; fue desarrollada y propuesta por el Dr. Enrnst Kussul y la Dra. Tetyana Baydyk (ver sección 4.6); se seleccionó sobre otros algoritmos debido a que posee las siguientes características de interés:

- Adaptable
- Robusto
- No utiliza operaciones de punto flotante
- Estructura en paralelo
- Implementación de bajo costo
- Alta velocidad de operación

Además de esto, el clasificador neuronal LIRA integra en su propia estructura la parte de extracción de características y codificación de la imagen (fig. 55), lo que hace que el diagrama de la figura 53 se simplifique aún más y estos tres bloques (E. de característica, caracterización y clasificación) se combinen en un solo bloqe llamado clasificador LIRA (Ver fig. 54)

Como se puede observar en la figura 54 el entrenamiento de la red y la determinación de sus paramentos óptimos (para esta aplicación) se realiza mediante un procedimiento externo al sistema, en otras palabras, la red neuronal tiene un aprendizaje fuera de chip (off-chip). Los parámetros obtenidos durante el entrenamiento y los

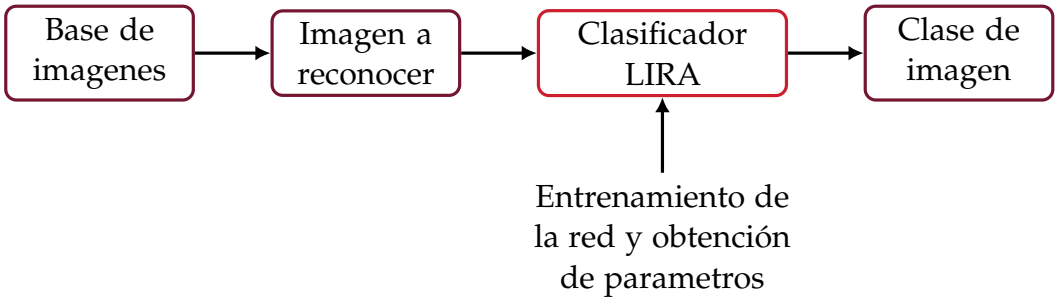


Figura 54: Diagrama de bloques del sistema de reconocimiento

pesos sinápticos entrenados son importados al sistema durante su implementación.

Expandiendo el bloque del clasificador LIRA (ver fig. 55) se puede observar que esta compuesto por cuatro partes principales que corresponden a las capas de la red neuronal LIRA-gray. Debido a que el entrenamiento de la red se realiza fuera del chip, esta debe contemplarse como parte del sistema total (sistema técnico Orfeo). Sin embargo, este último debe dividirse para hacer la distinción entre los dos procesos: entrenamiento e implementación en FPGA.

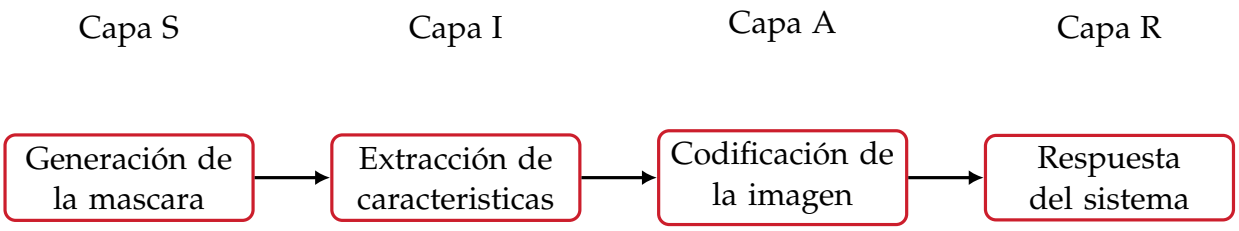


Figura 55: Diagrama de bloques del clasificador LIRA

En la figura 56, se muestra el diagrama con esta división del sistema técnico Orfeo en dos arreglos principales, denominados: N.1 y M.1. El primero, contempla el entrenamiento y el ajuste de parámetros mediante lenguaje de alto nivel de la red, de él se exportan los valores de los pesos sinápticos y los valores umbrales de las neuronas, así también, se determinan los parámetros óptimos para esta aplicación mediante experimentos realizados con simulaciones. El segundo arreglo, consiste en el diseño de la implementación del clasificador neuronal en FPGA; cada parte del algoritmo es validada mediante simulaciones con las herramientas de software para el análisis y la síntesis de diseños realizados en HDL.

Figura 56: Diagrama de las divisiones del sistema técnico Orfeo

Para mayor claridad, en la figura 57 se presenta diagrama de Venn donde se aprecia la intersección de los arreglos (N.1 ∪ M.1). Los pesos entrenados y los valores umbral obtenidos son el enlace entre ellos dos.

Figura 57: Intersección de los dos Arreglos

9.2 Arreglo N.1

No es más que el clasificador neuronal LIRA programado en Lenguaje de alto nivel, es entrenado mediante la base de imágenes PDS; como resultado se obtienen los parámetros óptimos de trabajo y se fijan como estructura final. Se realizaron pruebas de reconocimiento para determinar su desempeño y se exportaron los valores umbrales y los valores de los pesos sinápticos después del entrenamiento.

En la figura 58 se muestra el diagrama de bloques correspondiente al segmento N.1,

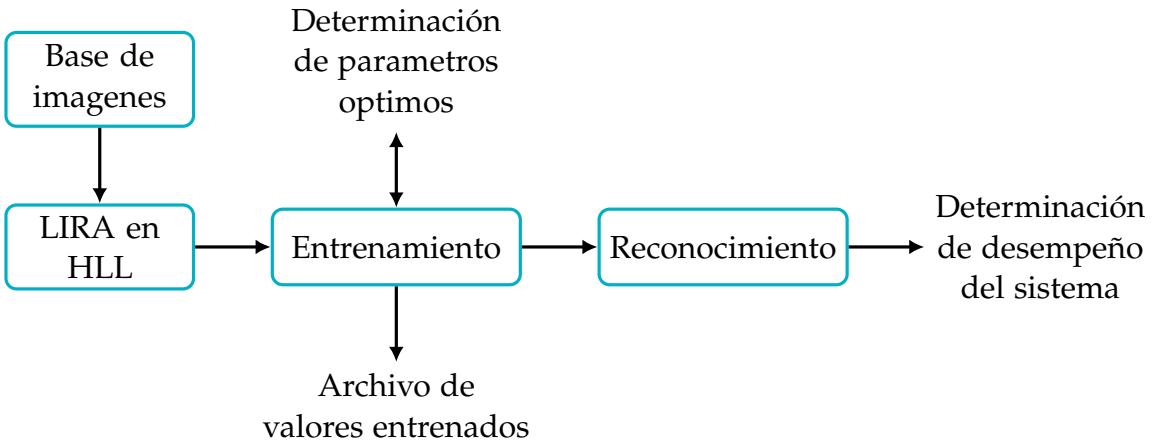


Figura 58: Diagrama de bloques del arreglo N.1

9.2.1 *Entrenamiento*

El entrenamiento es del tipo supervisado, el conjunto de entrenamiento está conformado de patrones de entrada con resultados previamente dados que le permiten a la red percibir conocer el error durante el reconocimiento. Aunque este procedimiento de aprendizaje no es uno de los mejores en términos de ser *biológicamente plausible*, es extremadamente efectivo y por lo tanto muy práctico. Por lo tanto, el proceso de aprendizaje realizado incluye los siguientes pasos:

1. **Introducción** del patrón de entrada (activación de neuronas de entrada)
2. **Propagación hacia adelante** de la entrada por la red, generación de la salida
3. Se **compara** la salida con la salida deseada (entrada de enseñanza), se proporciona el vector de error (vector de diferencia),
4. Las **correcciones** de la red se calculan en función del vector de error
5. La **corrección** es aplicada

9.3 Arreglo M.1

No es otra cosa que la estructura del clasificador neuronal LIRA descrita en hardware, reside en el FPGA. Los paramentos y valores de los pesos sinápticos residen en una memoria aparte y son importados al dispositivo; las imágenes a reconocer se transmiten al sistema mediante un protocolo de comunicación. El valor del diseño radica en la técnica elegida y desarrollada para describir redes neuronales en FPGA. En la figura 59 se muestra un diagrama de bloques del arreglo en general.

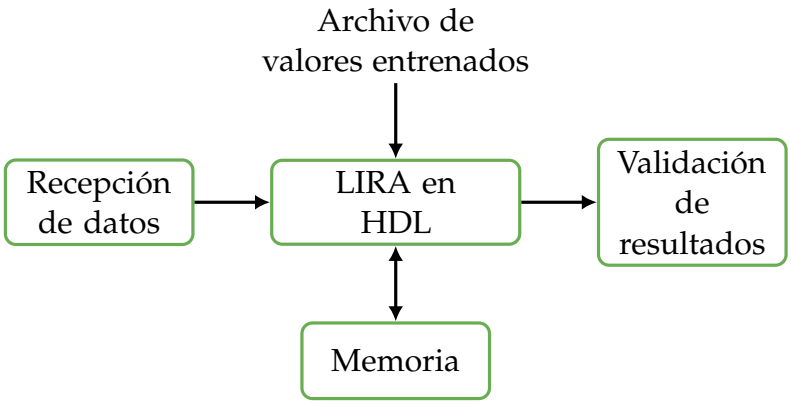


Figura 59: Diagrama de bloques del arreglo M.1

9.3.1 *Flujo de diseño*

El camino seguido en el diseño del sistema en FPGA toma como base el diagrama de flujo mostrado en la figura 60. El proceso comienza con la entrada del diseño mediante HDL o esquemas, continúa con la simulación para verificar que la lógica se implemente como se espera; después, se utilizan las herramientas de software para realizar la síntesis, para mapear la lógica a la arquitectura del dispositivo. A continuación, la herramienta creara la conexión entre las celdas colocando un enrutamiento o ajuste del diseño.

Posteriormente, se requiere de un análisis de tiempos utilizando herramientas de tiempo para asegurar que no haya infracciones de configuración o algún hueco en el diseño. También, se ejecutar otra simulación después de realizar el ajuste para así descubrir problemas de sincronización.

Después de que el análisis confirma el funcionamiento correcto del circuito, el siguiente paso es utilizar la herramienta para crear un archivo de programación, que luego se descarga en el dispositivo FPGA para hacer pruebas. El último bloque mostrado en el diagrama llamado *lanzamiento*, se refiere a que el producto está funcional y listo para comenzar los procesos de comercialización o integración a un proyecto más grande.

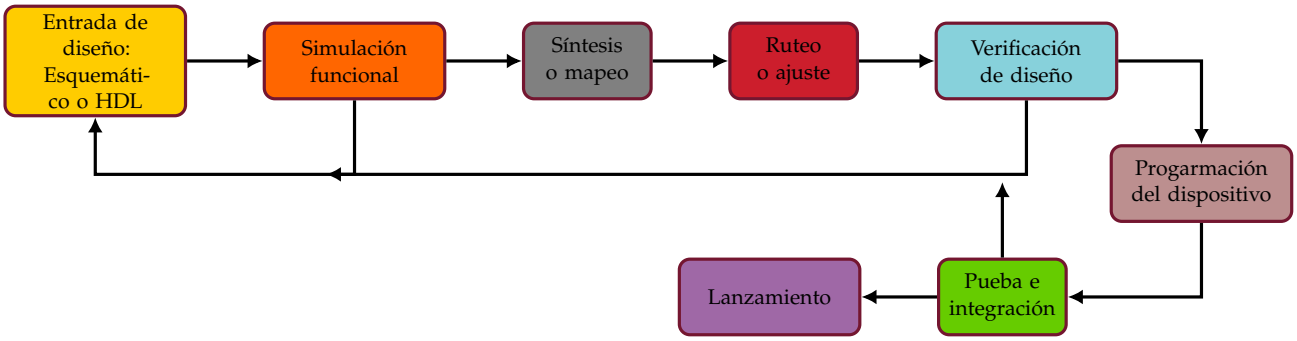


Figura 60: Flujo de diseño en FPGA

No obstante, el alcance de este trabajo no abarca los tres últimos bloques del diagrama de la figura 60, la verificación de los sistemas diseñados se da mediante pruebas y experimentos de simulación con la ayuda de la herramienta de software. Por lo tanto, el diagrama de flujo de diseño en FPGA utilizado en este trabajo se simplifica como se muestra en la figura 61.

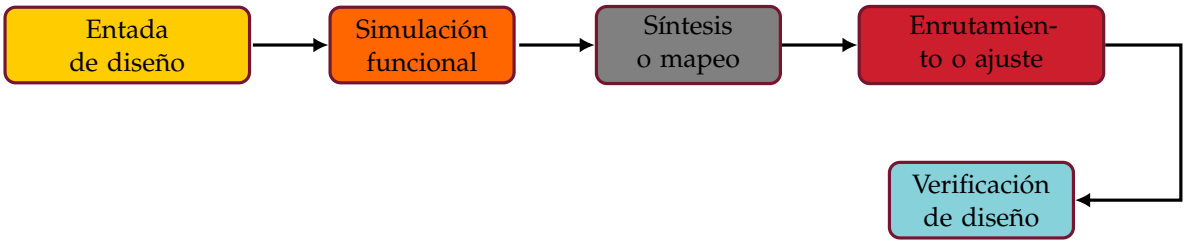


Figura 61: Flujo de diseño simplificado

En el bloque llamado *elemento de calculo* de la figura 59 se encuentra descrito el algoritmo del clasificador neuronal LIRA utilizando la metodología FPNA/FPNN.

Para facilitar el proceso de implementación, el funcionamiento de LIRA se visualiza mediante bloques funcionales, donde cada cada campa que lo conforma se designa de la siguiente manera:

S - Capa sensorial

I - Capa intermedia

A - Capa asociativa

R - Capa de Respuesta

9.4 Diseño LIRA-FPNA

Aplicar la metodología FPNA para implementar la red neuronal LIRA consiste en establecer los recursos FPNA para esta red en específico. En la figura 62, se puede observar la topología de la red LIRA totalmente conectada, por simplicidad solo se muestran las conexiones respectivas de una subventana hxw. donde se toman tantos pixeles de la capa S como neuronas ON/OFF existente en cada grupo de la capa I; cada uno de estos grupos se conecta con una neurona de la capa A, que a su vez esta tiene conexiones ponderadas con todas las neuronas de la capa S. Estas conexiones se repiten para cada una de las subventanas de la capa S.

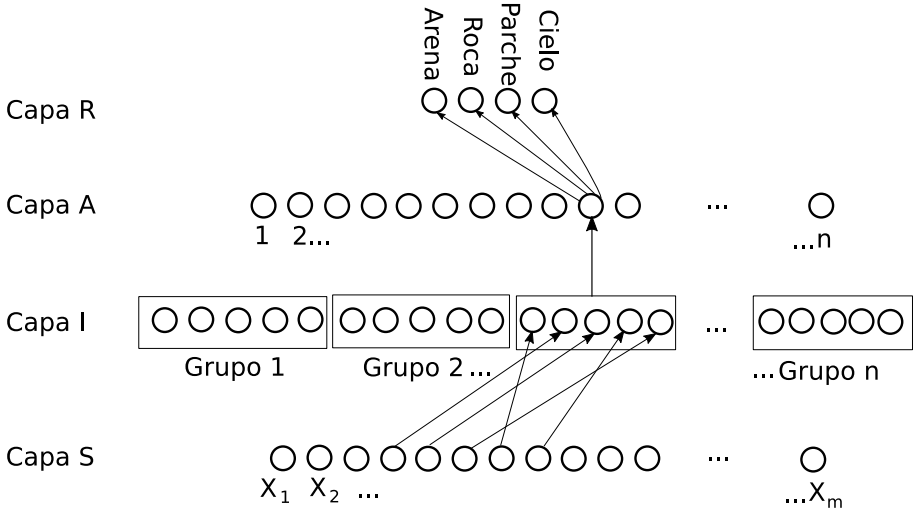


Figura 62: Red LIRA original

La idea es transformar la red totalmente conectada (fig. 62) en una red equivalente simplificada donde el número de conexiones se reduzca al generar procesos iterativos donde cada nodo es actualizado con nuevos valores de entrada.

Cómo consecuencia de esta transformación a cada nodo de la red simplificada se le deben asigna un conjunto de *activadores* [1] (j_i, f_i) y entre cada nodo un enlace que contiene un operador afín $\alpha_{p,n}$. Ver fig. 64.

Deacuerdo a la naturaleza de funcionamiento de la red LIRA, se sabe que en la capa (I) se tiene agrupaciones de neuronas que reciben valores de forma simultánea de la capa predecesora (S). Por lo tanto, en este diseño se considera conveniente establecer que el conjunto de neuronas que conforman un grupo (tanto de la capa (I), como de la capa (S)) forman un nodo en si de la nueva red, mientras que en las capas A y R cada nodo corresponderá a una neurona individual.

En la figura 65 se muestra lo dicho antes, en este caso, un nodo de la capa (I) contiene cinco neuronas, al igual que un nodo de la capa (S) que contiene cinco neuronas o píxeles de la imagen.

1 Se cambio la variable i por j en el activador de iteración para no causar confusión con los indices de la capa intermedia (I)

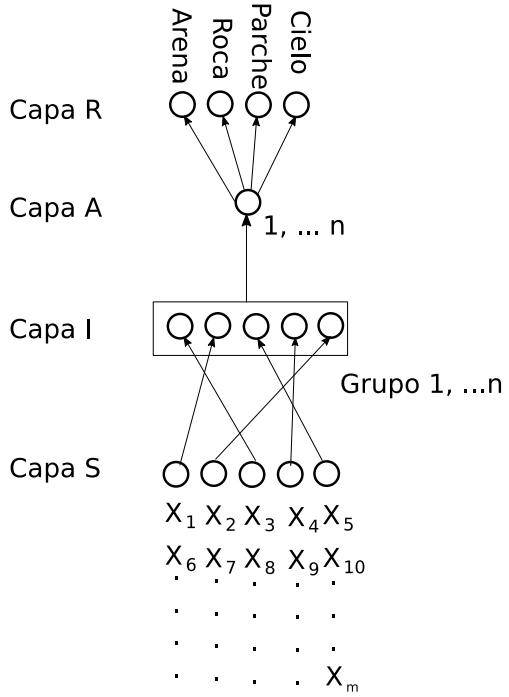


Figura 63: Red con menos conexiones y nodos actualizables

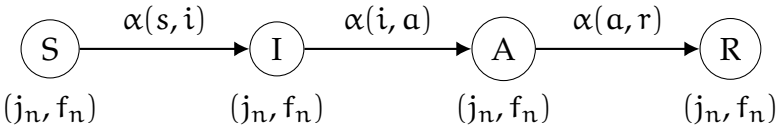


Figura 64: Recursos en LIRA

Entre la capa (I) y la capa (A) puede decirse que existe una correspondencia uno a uno (un bloque intermedio, una neurona asociativa) por lo que las operaciones aritméticas de estas dos capas puede agruparse como un solo bloque de calculo (BC) y así obtener una salida resultante para cada conjunto de de datos de entrada. Por lo tanto, se definen los siguientes tres bloques funcionales que conforman la topología LIRA-FPNA:

1. Bloque de entrada (Capa S)
2. Bloque de calculo (Capa I, Capa A)
3. Bloque de salida (Capa R)

El bloque de salida (capa R) recibe información de otro bloque que contiene los valores de los pesos sinápticos almacenados (Fig. 66). No obstante, los dos bloques anteriores también necesitan información adicional para su operación, se sabe que para el bloque de entrada (capa S) se requieren de las coordenadas absolutas de los píxeles a analizar, es decir, la máscara generada durante la etapa de entrenamiento y para el bloque de calculo (capas I, A), es necesario contar con los valores umbrales de cada una de las neuronas de la capa I. Por lo que es necesario contar bloques de memoria en los cuales se importen y almacenen los valores entrenados.

Así mismo, debido a la naturaleza de la red neuronal LIRA se sabe que los pesos sinápticos se multiplican por los valores del vector de características generado en

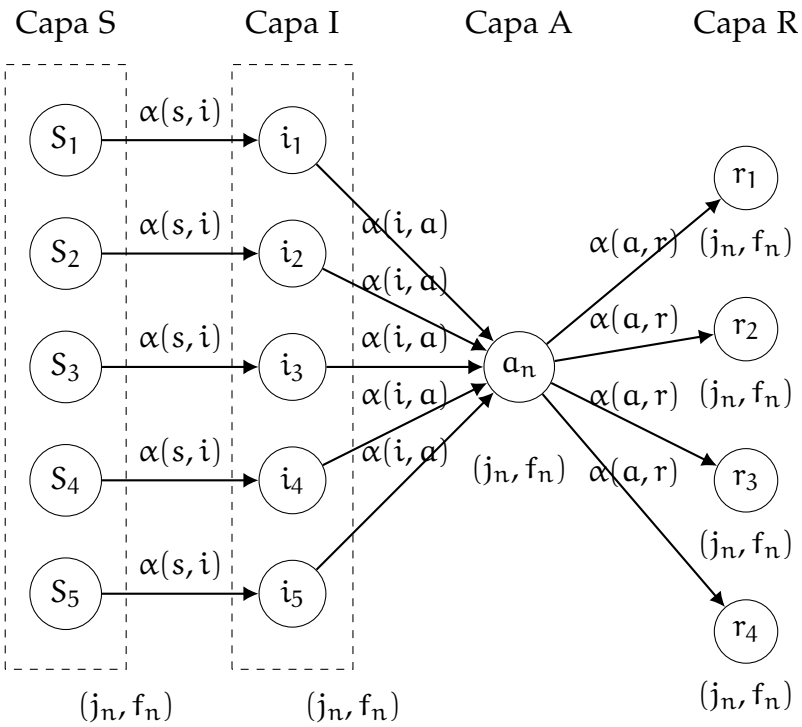


Figura 65: Relación de nodos con recursos FPNA

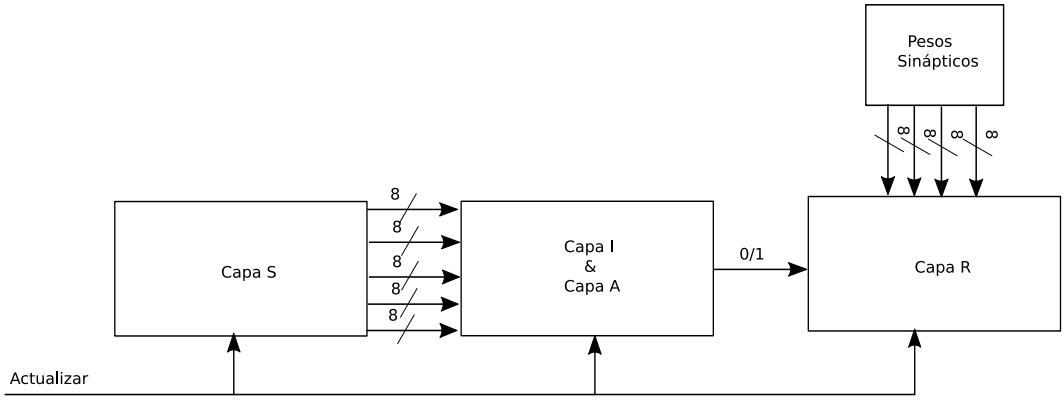


Figura 66: Diagrama de bloques para LIRA-FPNA

la capa A, este es un vector binario y los pesos se multiplican por uno o cero, lo que es lo mismo a establecer si dichos pesos aportan a la sumatoria final.

Dicho esto, el bloque correspondiente a la capa R se integra dentro del bloque de cálculo. En la figura 67 se muestra el diagrama modificado con los bloques principales para LIRA-FPNA con los bloques que almacenan los valores necesarios. Para resumir, el diseño del FPNA LIRA escala de grises consistió en determinar los bloques funcionales mostrados anteriormente (fig. 67), y en establecer los activadores (j_n, f_n) y los operadores afín[2] ($\alpha_{p,n}$), en otras palabras, establecer los recursos FPNA.

2 Tanto los activadores como los operadores afín serán desarrollados más adelante en el capítulo 11.

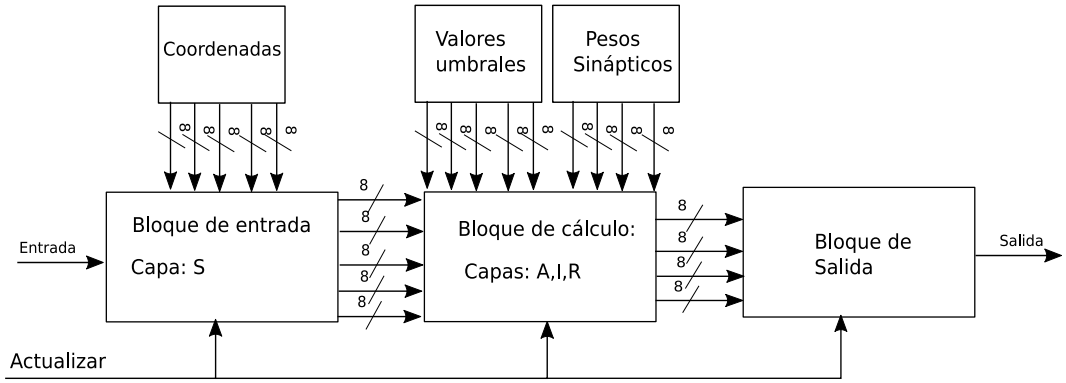


Figura 67: Diagrama de Bloques para LIRA-FPNA con memorias de datos

9.5 Diseño LIRA-FPNN

La distinción entre FPNA y FPNN está relacionada principalmente con las propiedades de implementación. Un FPNA corresponde a un conjunto dado de recursos neuronales que se organizan de acuerdo con relaciones de vecindad específicas, mientras que un FPNN es una forma de utilizar este conjunto de recursos: solo implica opciones de configuración local.

Se han definido varios métodos de cálculo para los FPNN. En cualquiera de estos métodos, todos los recursos se comportan de forma independiente. Cuando un recurso recibe valores, aplica su (s) operador (es) local (es) y envía el resultado a todos los recursos vecinos a los que está conectado localmente (un recurso neuronal espera los valores, antes de enviar una salida a sus vecinos).

La implementación del FPNA puede ser modular: se dan bloques predefinidos (recursos configurables) que realizan tanto cálculos como manejo de protocolo asíncronos, y se ensamblan de acuerdo con el gráfico FPNN.

El diseño LIRA-FPNN mostrado en la figura 68, es un esquema general para organizar los recursos LIRA-FPNA, en este esquema se describen múltiples bloques de cálculo que realizan operaciones de forma paralela gracias a su acomodo matricial, el cual puede ser tan grande como los recursos en el FPGA lo permitan.

Cada bloque de calculo opera sobre solo una parte de la red neuronal actualizando sus valores de entrada de forma iterativa, de esta forma se logra realizar un gran número de operaciones con ayuda de procesos iterativos en paralelo. Así mismo, la memoria que contiene los valores para el bloque de calculo se divide en múltiples bloques para alimentar a cada fila de la matriz. También, se agregan bloques adicionales de control y de selección de resultados.

El diagrama de bloques condensado del diseño de LIRA-FPNN mostrado en la figura 68 esconde mucha complejidad en los subsistemas que lo conforman, este es solo es un modelo general utilizado posteriormente en el diseño de detalle (capítulo 11) para generar la arquitectura final, es modificado según las necesidades y los recursos necesarios para implementar cada una de su partes.

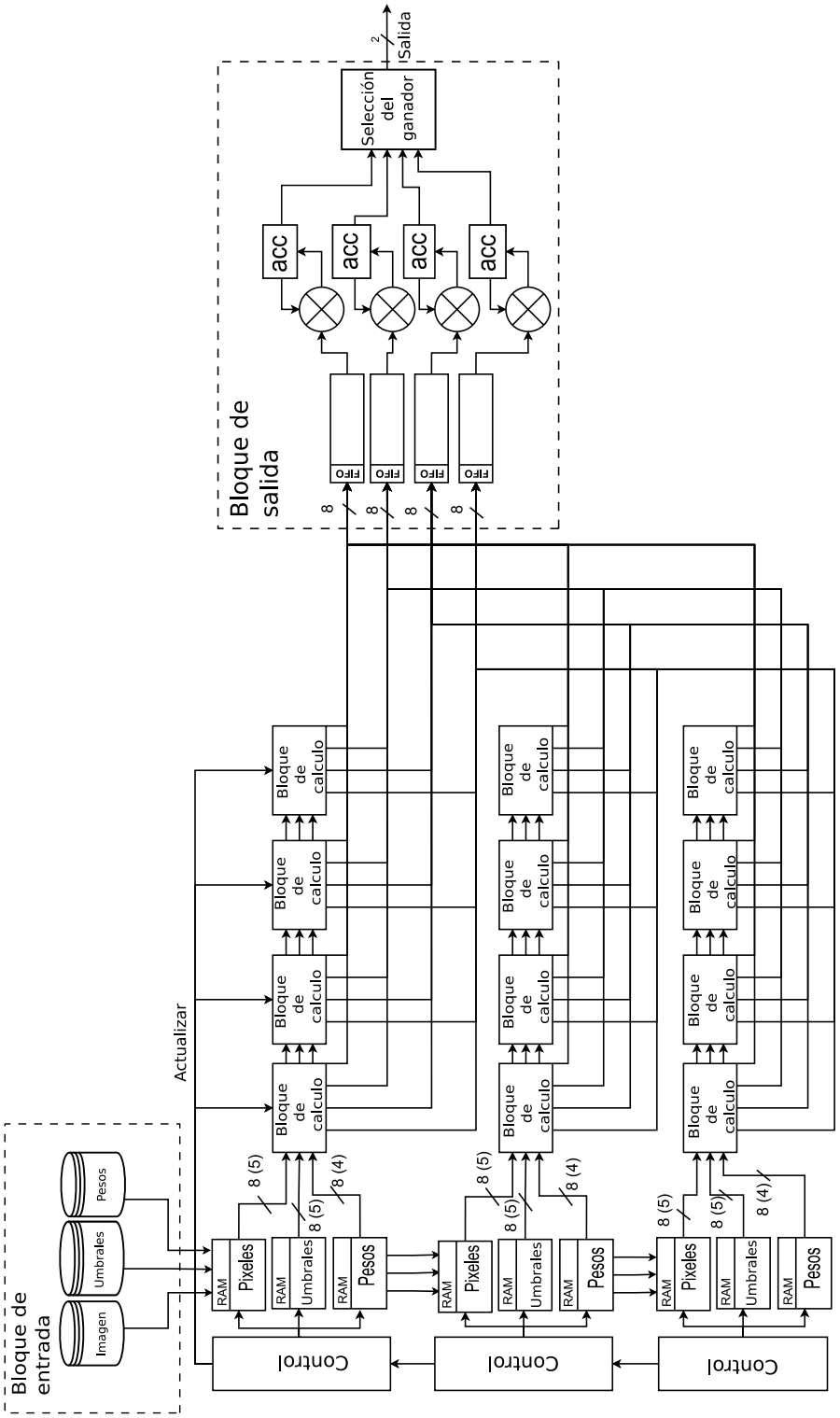


Figura 68: Diagrama FPNN general para LIRA

Parte III

DISEÑO DE DETALLE

En la naturaleza, no hay separación entre diseño, ingeniería y fabricación; el hueso lo hace todo.

— **Neri Oxman**

10

ARREGLO: N.1

En este capítulo se documenta el diseño de detalle del arreglo N.1, se explica en detalle cada una de las etapas, desde la selección de las imágenes y construcción de la base de datos, hasta los experimentos realizados durante la verificación de funcionamiento. También, se explican algunas modificaciones que el Dr. Ernst Kussul realizó al algoritmo del perceptrón de Rosenblant y que son utilizadas para el funcionamiento correcto del clasificador LIRA escala de grises.

Tomando como base el clasificador neuronal LIRA en escala de grises presentado en la §4.6, el desarrollo del arreglo N.1 se dividió en tres fases:

1. Programación de la estructura de LIRA y su metodología de entrenamiento mediante lenguaje de alto nivel.
2. Entrenamiento de la red utilizando la base de imágenes *ITM* y determinar los valores óptimos de sus parámetros, así como los valores entrenables (pesos sinápticos).
3. Validar desempeño del sistema para el reconocimiento de las texturas establecidas.

10.1 Programación de la red

El clasificador neuronal LIRA fue programado en lenguaje Python 3.8 utilizando el entorno de desarrollo integrado (IDE, por sus siglas en ingles) *Spyder* perteneciente a la interfaz gráfica de usuario (GUI) *Anaconda Navigator*, la cual puede gestionar de manera avanzada paquetes relacionados a la ciencia de datos y permite compilar Python en código de máquina para una ejecución rápida.

Se eligió utilizar el lenguaje Python para la programación de la red neuronal debido a que posee varias ventajas sobre otros lenguajes de programación y son de interés para esta aplicación, por ejemplo:

- Es un lenguaje orientado a objetos
- Posee un estilo flexible
- Es de código abierto
- Es multiplataforma
- Cuenta con una comunidad activa

Además, es un lenguaje muy utilizado en la actualidad para aplicaciones de aprendizaje automático y programación de redes neuronales, por lo que posee librerías especializadas muy útiles para el procesamiento de imágenes y creación de base de datos, así como algoritmos para redes neuronales de última generación; todo esto permite comprobar el funcionamiento de algoritmos y/o porciones de código en corto tiempo y agilizar el progreso de un proyecto en particular.

El programa desarrollado se corrió en una computadora portátil con un procesador Intel Core(TM) i7 @ 2.20GHz y memoria RAM de 8Gb.

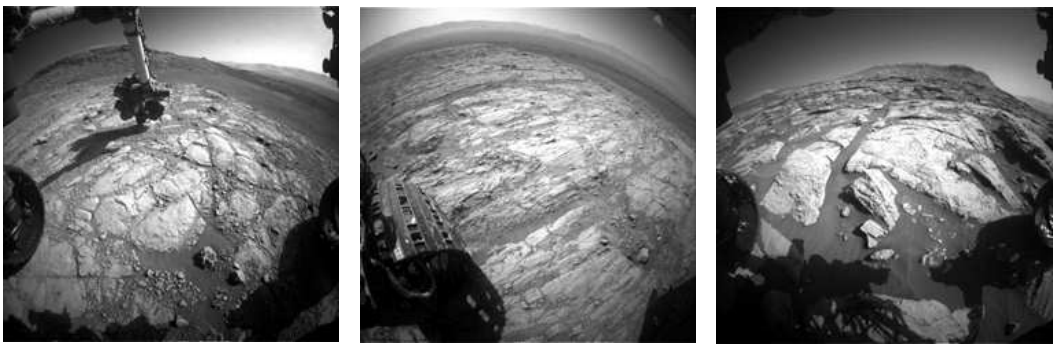
10.1.1 *Base de imágenes*

Para este trabajo se creo una base de imágenes a partir la bases de datos PIA y PDS descritas en la §1.4, se seleccionaron varias imágenes y se dimensionaron de acuerdo a los requerimientos, estas imágenes fueron utilizadas para el entrenamiento y experimentos y pruebas de reconocimiento de la red. Para fines de identificación, esta base de imágenes es designada con el nombre de: *imágenes del terreno marciano* ITM.

10.1.2 *Selección de las imágenes*

Los rover utilizados actualmente en la superficie de Marte contienen diferentes tipos de cámaras utilizadas para propósitos específicos (ver §3.3.1).

CÁMARAS DE INGENIERÍA: Brindan información del terreno al rededor del rover, se dividen en dos tipos: Cámaras de navegación (NavCams) y cámaras para evadir obstáculos (HazCams). En la figura 69 se muestran tres fotos tomadas por las cámaras HazCam, donde se aprecia que utilizan un lente ojo de pez, y por su ubicación, algunas partes del rover aparecen en la toma.



(a) (b) (c)

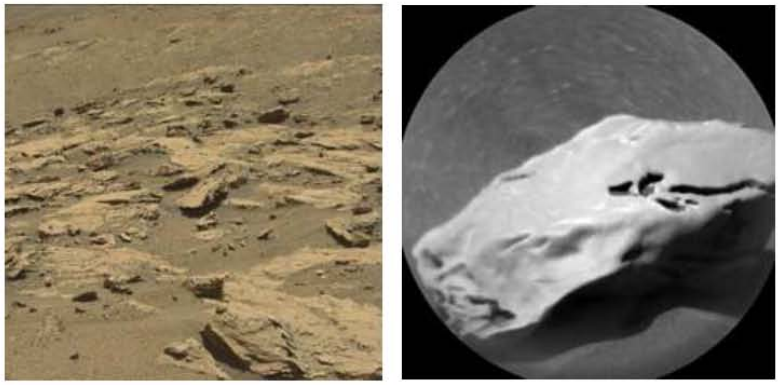
Figura 69: Imágenes de la cámara HazCam

También, en la figura 70 se muestran imágenes tomadas por las cámaras NavCam que poseen un arreglo óptico diferente y su ubicación permite tener una vista panorámica.

(a) (b) (c)

Figura 70: Imágenes de la cámara NavCam

CÁMARAS CIENTÍFICAS: Son cámaras especializadas para obtener información detallada de muestras del terreno marciano, están acompañadas de instrumentos con los cuales se puede analizar la composición elemental de las muestras y cuentan con arreglos ópticos capaces de realizar acercamientos muy potentes. La figura 71 muestra imágenes de las principales cámaras científicas, donde la MastCam se distingue porque utiliza imágenes a color.



(a) MastCam (b) ChemCam

Figura 71: Imágenes de las cámaras científicas

Las imágenes elegidas para formar parte de la base ITM pertenecen principalmente a las cámaras NavCam, ya que estas cámaras son capaces de proporcionan una gran variedad de imágenes tanto cercanas (fig. 72), como del horizonte (fig. 73), sin distorsiones ópticas relevantes. Así mismo, se descartaron imágenes que incluyen objetos o sombras que contaminen la escena, dificultando así el reconocimiento de la textura. (ver fig. 74).

(a) (b) (c)

Figura 72: Imágenes cercanas

(a)

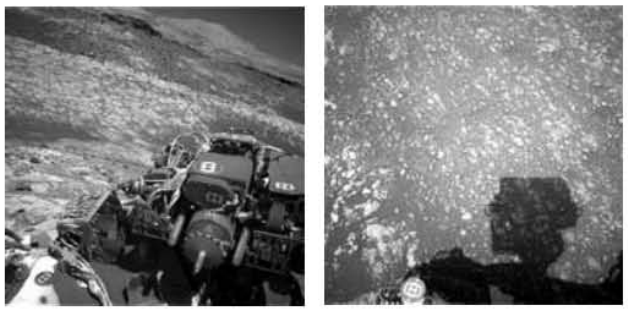
(b)

Figura 73: Imágenes del Horizonte

De las imágenes elegidas se escogieron los tipos de texturas más sobresalientes en la escena, estas seis texturas pueden apreciarse la figura 75, y son: arena de grano medio, arena de grano fino, parche de roca sobre arena, terreno rocoso, cielo y terreno accidentado, donde este último es una combinación de los anteriores.

Como un primer diseño, se decidió trabajar con cuatro clases a reconocer, debido a que se desconoce en principio los recursos totales necesarios para implementar la red neuronal. Por lo tanto, los seis tipos de terreno antes mencionados fueron reducidos a cuatro al desaparecer la distinción entre las dos categorías de arena y descartando la clase donde los terrenos se encuentran combinados. Dicho lo anterior, las clases establecidas a reconocer son las siguientes:

(a) Rueda (b) Dos ruedas



(c) Chasis (d) Sombra

Figura 74: Imágenes con partes y sombras del rover

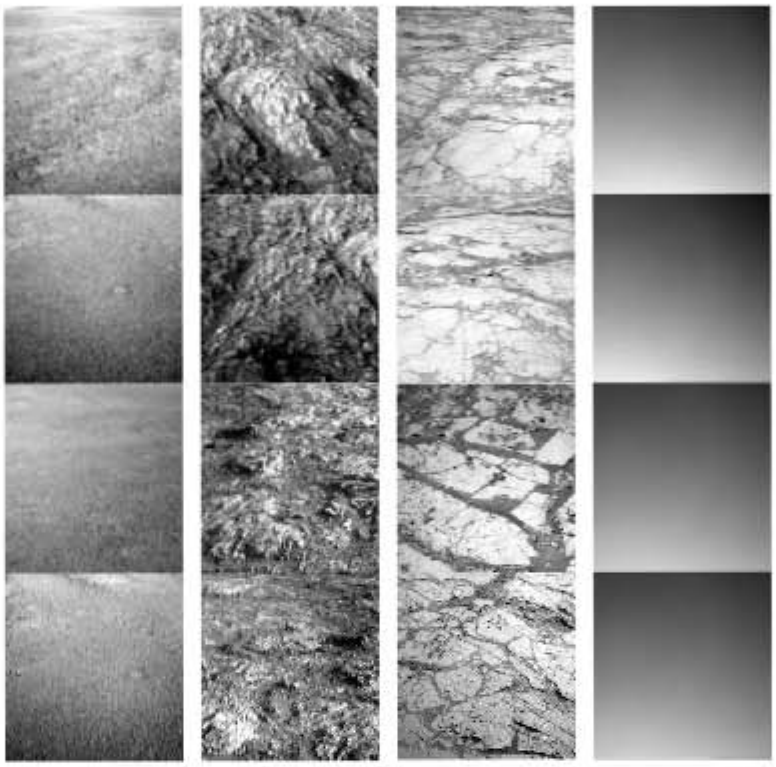
(a) Grano medio (b) Grano fino (c) Parche

(d) Roca (e) Cielo (f) Accidentado

Figura 75: Imágenes con partes del rover

1. **Arena** - Superficie *plana* sin ningún obstáculo geométrico
2. **Roca** - Terrenos rocoso y obstáculos relativamente pequeños
3. **Parche** - Superficie compuestas de grandes rocas inmersas en la arena
4. **Cielo** - Degradado liso en escalas de grises

En la figura 76 se muestran cinco imágenes de cada clase, iguales a las que se utilizan en el clasificador neuronal LIRA.



(a) Arena (b) Roca (c) Parche (d) Cielo

Figura 76: Las cuatro clases a reconocer

Las imágenes originales de las cámaras NavCam se encuentran en escala de grises y tienen dimensiones de 1024 × 1024 píxeles, para construir la base ITM se escanearon estas imágenes en ventanas con dimensiones de 256 × 256 píxeles, la base ITM contiene un total de 200 imágenes (50 imágenes para cada clase), Estas 200 imágenes son guardadas en una carpeta por clase de donde se extraen para ser procesadas.

Con la base ITM creada se continuó con la programación en lenguaje de alto nivel del arreglo N.1, basado en el clasificador neuronal LIRA, para el reconocimiento de imágenes de terreno marciano.

10.1.3 *Capa sensorial: S*

La capa sensorial (S), también llamada retina, es la capa donde se ingresa la imagen a reconocer, la cual es seleccionada de la base ITM y tiene las características mostradas en la tabla 20. En esta capa, la imagen de entrada se escanea de manera aleatoria mediante subventanas y se generan conexiones aleatorias con la capa asociativa (A), a este proceso se le conoce como generación de la máscara, dicho de otra forma, la máscara contiene la información de como se va escanear la imagen y como serán las conexiones entre la capa S y la A.

Tabla 20: Especificaciones de la imagen de entrada

Alto (H) [px]	Ancho (W) [px]	Color	Profundidad de bits [bits]
256	256	Escala de grises	8

10.1.3.1 *Generación de la máscara*

La máscara del clasificador neuronal LIRA, corresponde al número de conexiones positivas y negativas de la capa (A) con la retina generadas de manera aleatoria; en el caso particular de LIRA en escala de grises estas conexiones se realizan mediante de la capa intermedia (I) y corresponden al número de conexiones ON/OFF.

MODELO MATEMÁTICO Para comenzar, la imagen a reconocer($H_s \times W_s$) se escanea mediante subventanas con dimensiones ($h \times w$). Ese proceso comienza con la selección aleatoria de la esquina superior izquierda de la subventana utilizando las distancias dx y dy (fig. 77).

$$\begin{aligned} dx_i &= random_i(W_s - w) \\ dy_i &= random_i(H_s - h) \end{aligned} \tag{10.4}$$

Donde i es la posición de la neurona en la capa asociativa A y su correspondiente bloque en I, y $random_i(z)$ es el número aleatorio que se distribuye uniformemente en el rango $[0, z]$. A continuación, la posición de cada una de las conexiones positivas y negativas dentro del la subventana se define por:

$$\begin{aligned} x_{ij} &= random_{ij}(w) \\ y_{ij} &= randomij(h) \end{aligned} \tag{10.5}$$

donde j es el número de la i-ésima conexion neuronal con la retina. Por lo tanto las coordenadas absolutas de las conexiones en la retina se definen como:

$$\begin{aligned} X_{ij} &= x_{ij} + dx_i \\ Y_{ij} &= y_{ij} + dy_i \end{aligned} \tag{10.6}$$

10.1.3.2 *Generador de números aleatorios*

El funcionamiento del clasificador neuronal LIRA requiere generar una gran lista de números pseudo aleatorios, estos son utilizados diferentes tareas en el clasificador. No obstante, aunque Python contiene una biblioteca especializada para generar números aleatorios, no se tiene un completo control sobre el proceso que hay detrás para generar dichos números, y la generación de la misma cadena de números en otro lenguaje puede dificultarse. Por lo tanto, se considerar la opción de programar un generador de números aleatorios propio.

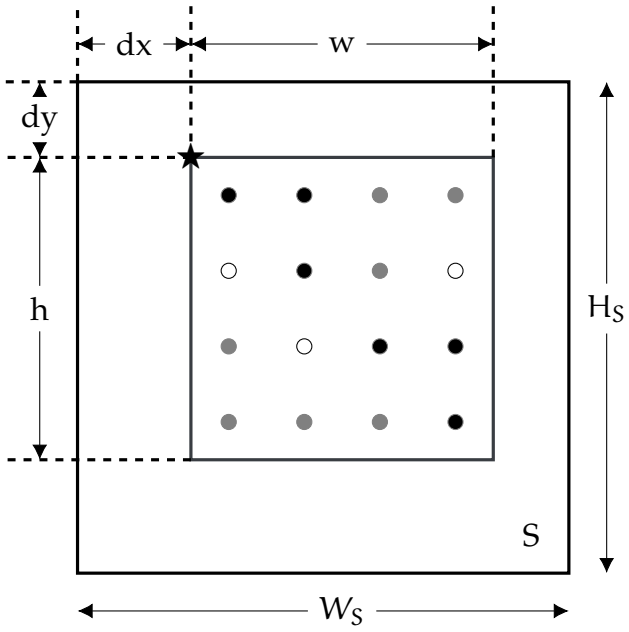


Figura 77: Parámetros de la imagen en S

Dicho lo anterios, se eligió programar un generador lineal congruencial (GLC), el cual es uno de los métodos más antiguos y más estudiados para la generación de números aleatorios (ver anexo E). Para configurarlo se utilizaron los parámetros de Borland C/C++ mostrados en la tabla 34.

Listing 1: GLC en Python

```
        def seedLCG(initVal):
          global rand
          rand = initVal

        def lcg(x,y):
          a = 22695477
          c = 1
          m = 2**32
          global rand
          rand = ((a*rand + c)  % m) % y + x
          return rand

    seedLCG(19)
```

Se realizó un experimento para observar y comparar el tiempo que tarda cada uno de los métodos disponibles para la generación de números pseudoaleatorios (biblioteca random y GLC).

El experimento consistió en generar 32000 números aleatorios diez veces y obtener el tiempo mínimo, el tiempo máximo y el tiempo promedio para generar esta cadena. También, se observo el comportamiento de los números y se verificó que el periodo fuera menor al numero total de cifras generadas.

Listing 2: Utilizando la biblioteca random

```
import time
tiempoIn1 = time.time()
import random
random.seed(19)
for i in range(32000):
rand1 = random.randint(0,10)
print(rand1)
tiempoFin1 = time.time()
print("Tarda: ",tiempoFin1 - tiempoIn1, "segundos")
```

Listing 3: Programando el GLC

```
import time
tiempoIn2 = time.time()
seedLCG(19)
for j in range(32000):
rand2 = lcg()
print(rand2)
tiempoFin2 = time.time()
print("Tarda: ",tiempoFin2 - tiempoIn2, "segundos")
```

Los resultados para el tiempo de las diez corridas se muestra en la tabla 21. Se puede apreciar que el tiempo para el GLC programado es incluso mejor que utilizar la biblioteca random, y cuenta con la ventaja de poder ser implementado en FPGA eficientemente.

Tabla 21: Tiempo de procesamiento

Método	Tiempo mínimo	Tiempo Máximo	Tiempo promedio
Biblioteca random	0.0154 [s]	0.0437 [s]	0.02093 [s]
Generador lineal congruencial	$< 1[\mu s]$	0.0176 [s]	0.00864 [s]

10.1.4 *Capa intermedia: I*

La capa intermedia (I), como su nombre lo indica sirve como intermediario entre las conexiones de la retina (S) y la capa asociativa (A), la entrada de cada neurona en la capa (I) tiene una conexión con la capa (S) y su salida está conectada con la entrada de una neurona en la capa (A). Las neuronas en la capa intermedia se dividen en grupos, donde el número de neuronas en un grupo corresponde al número de conexiones positivas y negativas entre una neurona de la capa (A) y la retina. Así mismo, existen dos tipos de neuronas en la capa (I): neuronas ON y neuronas OFF, donde su comportamiento es el siguiente:

10.1.4.1 *Modelo matemático*

$$\Psi_{ON}(x_i) = \begin{cases} 1, x_i \geqslant \theta_i \\ 0, x_i < T_i \end{cases} \tag{10.7}$$

$$\Psi_{OFF}(x_j) = \begin{cases} 1, x_j \leqslant \theta_j \\ 0, x_j > T_j \end{cases}$$

Donde Ψ es el valor de salida de las neurona ON/OFF, (x_i,x_j) son los valores de entrada de las neuronas ON/OFF y (θ) es su valor umbral, la salida de las neuronas ON es 1 cuando su valor de entrada es mayor o igual que el umbral θ_i, y su salida es 0 si resulta ser menor. Por otro lado, la salida de las neuronas OFF es 1 cuando su entrada es menor o igual que su valor umbral θ_j, y sera 0 si resulta ser mayor.

Los umbrales T_i y T_j se seleccionan nuevamente de forma aleatoria en el rango $[0, \eta \cdot b_{max}]$, donde b_{max} es el brillo máximo de los píxeles de la imagen, y η es un parámetro que se selecciona de manera experimental y se encuentra en el rango $[0, 1]$.

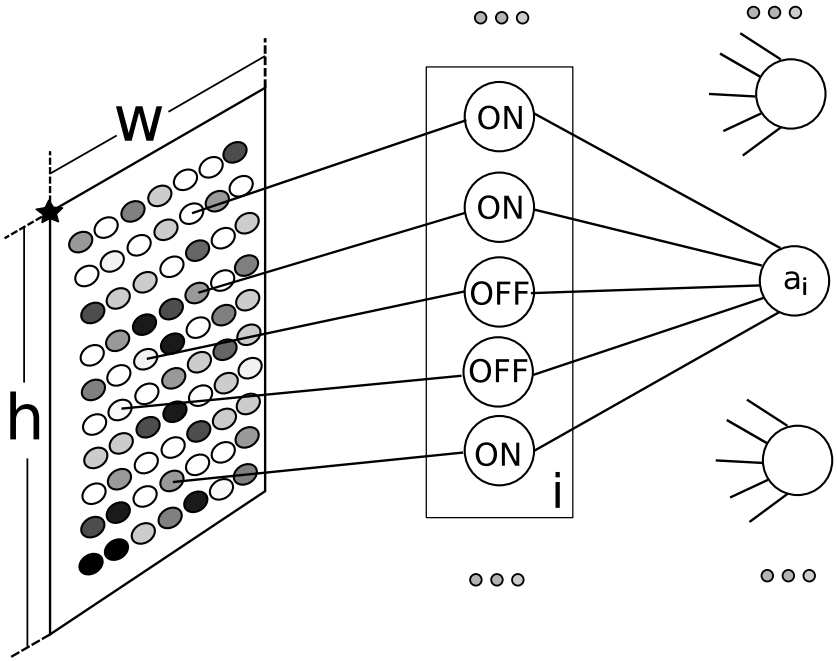


Figura 78: Conexiones de la capa intermedia

En la figura 78 se muestra un driagrama con las conexiones de una subventana, un bloque de la capa (I) y una neurona de la capa (A). También en la figura 79 se muestra una captura de pantalla donde se pueden apreciar los niveles de brillo de los píxeles que conforman la imagen.

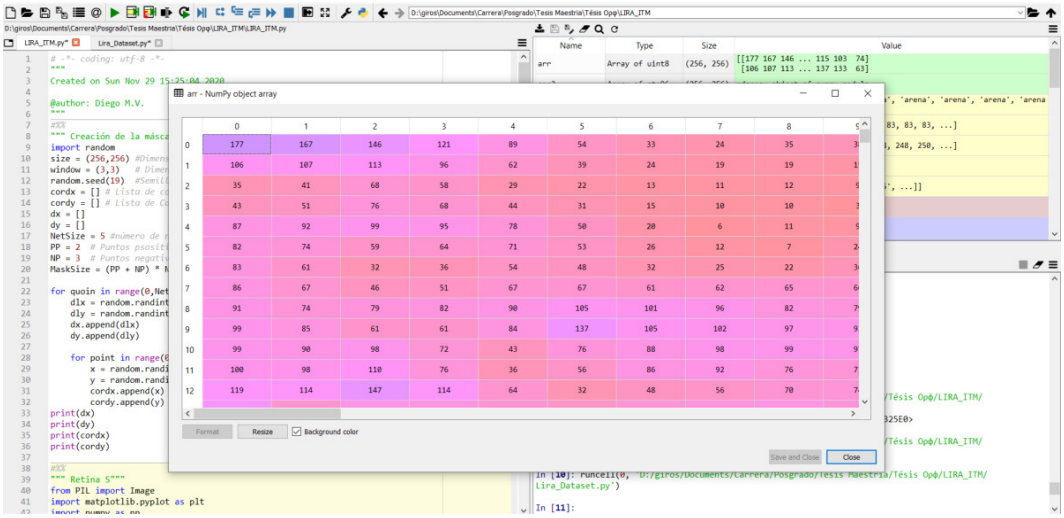


Figura 79: Matriz con niveles de brillo de los pixeles

10.1.5 *Capa asociativa : A*

Las entradas de cada neurona en la capa asociativa tienen correspondencia uno a uno con cada grupo de la capa intermedia; la salida de cada neurona en (A) será 1 si todas sus entradas están activas (1); si al menos una neurona del grupo que conecta a la neurona en (A) esta inactiva (0), la neurona asociativa tendrá un valor de salida igual a cero, en otras palabras, cada neurona de la capa asociativa se comporta como una compuerta lógica AND. El vector binario que representa la actividad de todas las neuronas asociativas se denomina código binario de la imagen, $A = a_1 ..., a_n$, (donde n es el número de neuronas en la capa A). Por lo tanto, el procedimiento que transforma la imagen de entrada en el vector binario A se denomina codificación de la imagen. En [44] se explica que se sabe por experiencia que el número de neuronas activas m en la capa (A) debe ser muchas veces menor que el número total de neuronas n de esta capa. Se utiliza entonces la siguiente relación:

$$m = c\sqrt{n} \tag{10.8}$$

Donde c es una constante en el rango $[1, 5]$. Esta relación corresponde a hechos neurofisiológicos, donde el numero de neuronas activas en la corteza cerebral es cientos de veces menor que el número total de neuronas[44].

Teniendo en cuenta lo anterior y sabiendo que el vector binario contiene una gran cantidad de ceros, mucho mayor al número de unos, resulta conveniente representarlo como una lista de números de neuronas activas en lugar de explícitamente. Por ejemplo, si el vector binario es igual a:

$$A = 00001000001000011000$$

La lista correspondiente del número de neuronas activas sera: [5,11,16,17], que se utiliza para guardar los códigos de la imagen en forma compacta y realizar un cálculos rápidos en el proceso de entrenamiento. Para este trabajo se tienen cinco neuronas en cada grupo de la capa (I), dos neuronas ON y tres neuronas OFF.

10.1.6 *Capa de respuesta: R*

La capa de respuesta establece la salida del sistema, esta capa tiene tantas neuronas como clases a reconocer, en nuestro caso cuatro. Cada neurona de la capa asociativa tiene conexiones ponderadas con cada neurona de la capa (R). La excitación E_i de cada neurona en la capa de (R) se define como:

$$E_i = \sum_{j=1}^{n} a_j * w_{ji} \tag{10.9}$$

Donde E_i es la excitación de la i-ésima neurona de la capa R, a_j es la excitación de la j-ésima neurona de la capa A; y w_{ji} es el valor de peso de la conexión entre la j-ésima neurona de la capa A y la i-ésima neurona de la capa R. Al final la neurona de la capa R que tenga la excitación más alta determinara la clase reconocida.

10.2 Entrenamiento de la red

El funcionamiento del segmento N.1 se divide en dos fases: la primera es el proceso de entrenamiento de la red neuronal utilizando una parte de la base de imágenes ITM. El método de entrenamiento de la red es del tipo supervisado, esto significa que para que la red neuronal aprenda a reconocer cada clase de imagen, un agente externo (supervisor o maestro) tiene que enseñarle o mostrarle la respuesta correcta que debe proporcionar ante dicha entrada.

10.2.1 *Algoritmo de entrenamiento*

El algoritmo para el entrenamiento supervisado se puede resumir de la siguiente manera: se presenta una imagen a la entrada del clasificador neuronal LIRA. Luego, el algoritmo calcula el código de la imagen y este es procesado para obtener una respuesta de salida; si la respuesta no corresponde a la clase correcta entonces los pesos sinápticos son modificados, este proceso se repite durante un número determinado de ciclos. Los pasos del proceso descrito anteriormente se enumeran a continuación:

PASO 1 Se inicializan los pesos sinápticos con un valor cercano a cero

PASO 2 La imagen se presenta a la entrada de la capa S

PASO 3 Se codifica la imagen mediante la capa I.

PASO 4 Generación del vector de características en la capa A

PASO 5 Cálculo de las sumas ponderadas entre la capa A y R

PASO 5 Respuesta del sistema en la capa de salida R

PASO 6 Modificación de los pesos W_{ij} de acuerdo según la respuesta

PASO 7 Repetir los pasos del 2 al 6 hasta alcanzar el número de ciclos establecidos

.

A continuación se muestra el diagrama de flujo del proceso de entrenamiento:

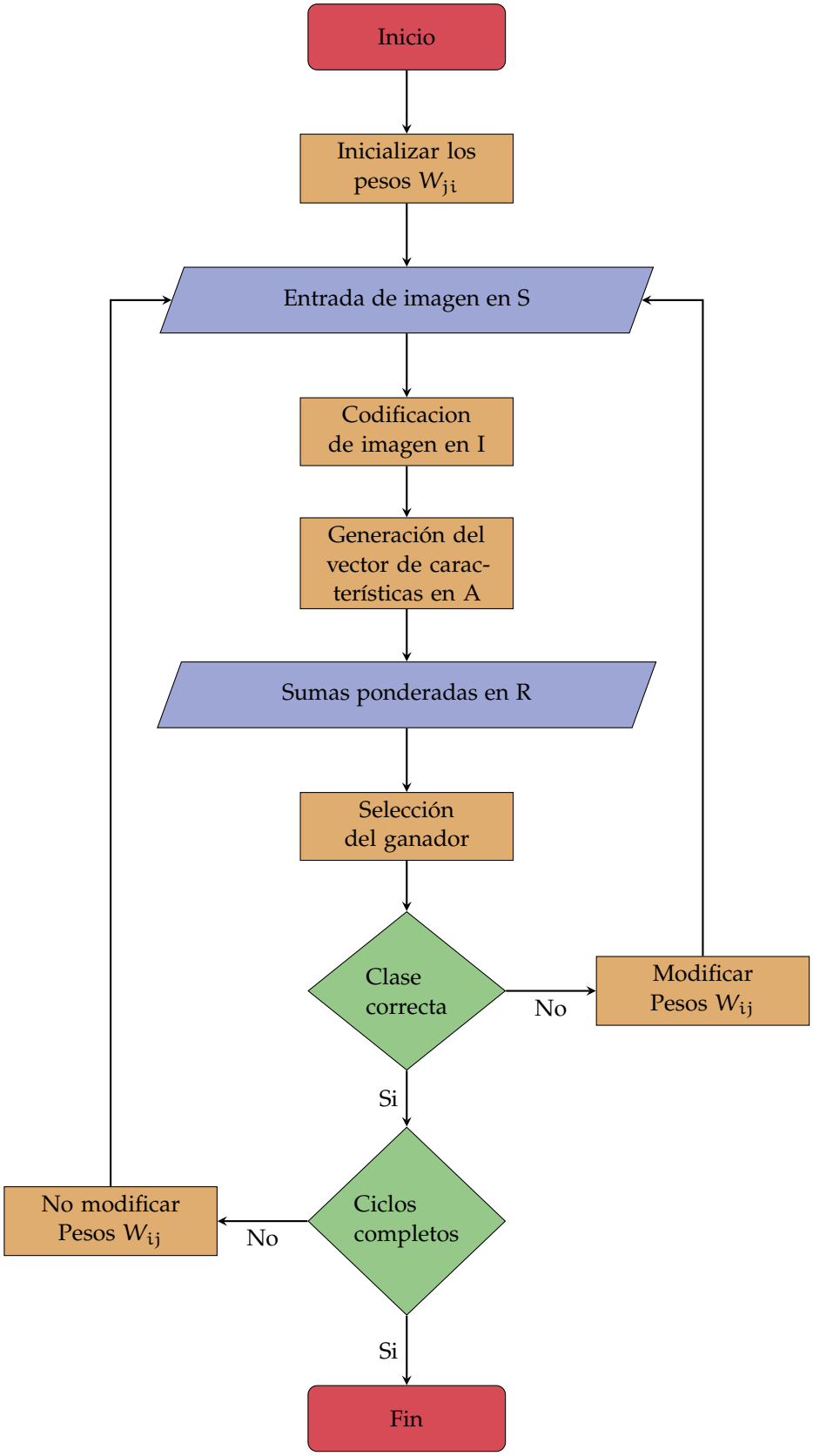


Figura 80: Diagrama de flujo del proceso de entrenamiento

10.2.2 *Selección del ganador*

Una de la modificaciones al perceptrón de Rosenblatt presentadas en [51] es para proceso de entrenamiento. En el clasificador neural LIRA, la neurona de la capa R que tiene la mayor excitación determina la clase bajo reconocimiento, esta regla se utiliza siempre en el proceso de reconocimiento. Sin embargo, en el proceso de entrenamiento esta regla cambia. Teniendo en cuenta que la neurona ganadora en la capa R tiene un valor de excitación E_w y su competidora más cercana tiene un valor E_c, si y solo si:

$$\frac{(E_w - E_c)}{E_w} < T_E \tag{10.10}$$

Entonces la neurona competidora se establece ahora como ganadora, donde T_E es la excitación superflua de la neurona ganadora.

Otra de las modificaciones es con respecto a las conexiones, las conexiones entre la capa A y la capa R de un perceptrón de Rosenblatt pueden ser negativas o positivas. No obstante, el clasificador LIRA solo utiliza conexiones positivas; en este caso, la regla de entrenamiento es la siguiente:

Sea j la clase correcta bajo reconocimiento, durante el proceso de reconocimiento

1. Se obtienen valores de excitación de las neuronas de la capa R.
2. La excitación de la neurona R_j (correspondiente a la clase correcta) se reduce por el factor $(1 - T_E)$
3. La neurona que tenga excitación máxima R_k se selecciona como ganadora.
4. Si $j = k$, no hay nada que hacer.
5. Si j no es igual a k, entonces:

$$w_{ij}(t+1) = w_{ij}(t) + a_i \tag{10.11}$$

Donde $w_{ij}(t)$ es el peso de la conexión entre la i-ésima neurona de la capa (A) y la j-ésima neurona de la capa R antes de la modificación, $w_{ij}(t+1)$ es el peso después de la modificación y a_i es la señal de salida (cero o uno) de la i-ésima neurona de la capa (A).

$$\begin{aligned} &w_{ik}(t+1) = w_{ik}(t) - a_i; && si\,(w_{ik}(t) \neq 0) \\ &w_{ik}(t+1) = 0; && si\,(w_{ik}(t) = 0), \end{aligned} \tag{10.12}$$

Donde $w_{ik}(t)$ es el peso de la conexión entre la i-ésima neurona de la capa A y la k-ésima neurona de la capa R antes de la modificación, y $w_{ik}(t+1)$ es el peso después de la modificación.

También en [44] se sugiere guardar la codificación binaria de cada imagen para usarlo en ciclos sucesivos del proceso de entrenamiento en lugar utilizar la misma imagen de entrada; esto da la posibilidad de ahorrar significativamente los recursos informáticos y el tiempo.

El proceso de entrenamiento se lleva a cabo de forma iterativa, una vez presentadas todas las imágenes del conjunto de datos, se calcula el número total de errores. Si este número es mayor que el uno por ciento del número total de imágenes, entonces se realiza el siguiente ciclo de entrenamiento, de lo contrario se detiene el proceso.

El proceso de entrenamiento también se detiene si el número de ciclos de entrenamiento realizados supera un valor predeterminado. Para la base de datos ITM este valor es 100.

10.2.3 *Conjunto de datos*

Previamente a la etapa de entrenamiento es conveniente construir un *conjunto de datos* con las imágenes seleccionadas para el entrenamiento, este corresponde a un archivo que contiene la información de cada imagen junto como su respectiva etiqueta (clase a la que pertenece); este conjunto de datos permite que la información de entrada fluya de manera continua y eficiente durante el entrenamiento.

Los pasos para generar el conjunto de datos en Python con las imágenes destinadas para el entrenamiento son los siguientes:

i Utilizar como parámetro de entrada la carpeta que contiene las imágenes con diferentes clases (guardar cada clase en una carpeta independiente).

ii Convertir cada imagen en una matriz Numpy con tipo de datos float32

iii Normalizar la matriz de imagen para tener valores entre 0 y 1, con esto se tiene una distribución de datos similar, y se logra una convergencia más rápida.

iv Marcar cada imagen con su respectiva etiqueta (la etiqueta se toma del nombre de la carpeta)

En la figura 81 se muestra una captura de pantalla del proceso para crear el conjunto de datos con las imágenes del terreno marciano.

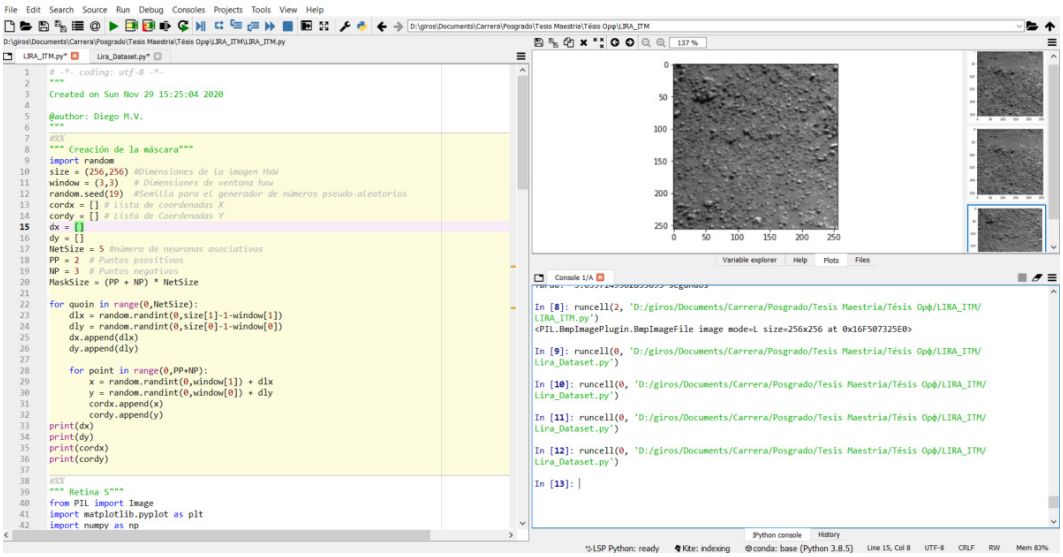


Figura 81: Creación del conjunto de datos para las clases a reconocer

A continuación se muestra (fig. 83) el conjunto de datos final, el cual es un arreglo que contiene todas las imágenes con formato Numpy float32 en cada una de sus filas.

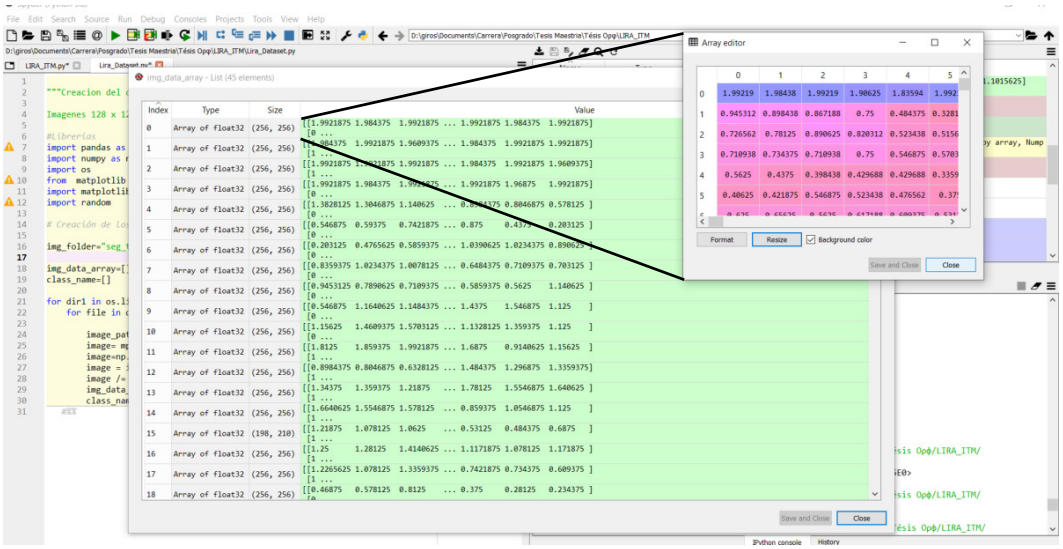


Figura 82: Conjunto de datos de las clases de terreno

También se tiene un arreglo adicional con las etiquetas de cada clase (fig. 83) estas etiquetas corresponden uno a uno con las matrices del conjunto de datos de la figura 83.

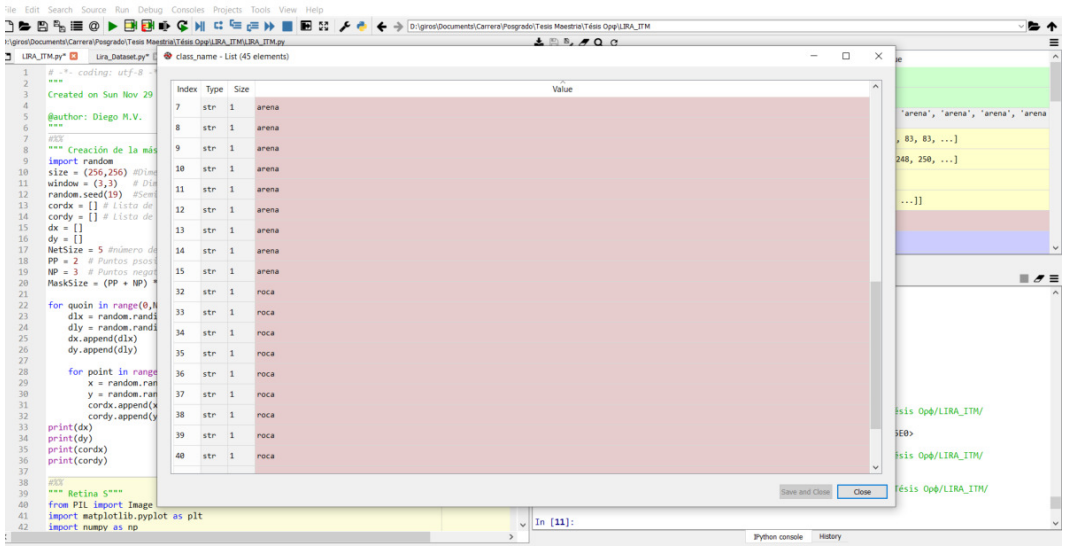


Figura 83: Arreglo con las etiquetas de cada clase

10.3 Reconocimiento

La segunda fase del segmento N.1 es la fase de reconocimiento, la neurosimulación consiste en dos fases: aprender y el recordar; en la fase de aprendizaje un grupo de muestras de entrenamiento son presentadas a la red y los pesos son ajustados de acuerdo a los valores de entrada, su correlación y los valores de salida, en otras palabras, la red aprende a resolver el problema; Mientras que en la fase de *memorizar*, también llamada de reconocimiento, una nueva entrada es presentada y la red eventualmente debe producir un resultado apropiado. Esta segunda fase sirve como verificación del funcionamiento del sistema y para estimar la efectividad del método., para esto, el parámetro más importante es la *tasa de reconocimiento*, este parámetro representa la relación entre muestras correctamente reconocidas y en las que el sistema se equivocó (errores).

10.3.1 *Experimentos*

Se realizaron experimentos preliminares para estimar el desempeño del segmento N.1, sobre la base de estos experimentos se seleccionaron los parámetros del clasificador que arrojaron los mejores resultados de reconocimiento. Posteriormente se realizaron experimentos finales para obtener la tasa de reconocimiento máxima. Los experimentos seleccionados se basaron en experimentos de artículos publicados sobre el clasificador neuronal LIRA:[38, 39, 51]

En el primer experimento preliminares, se incrementó gradualmente el número de neuronas de la capa asociativa de 2000 a 250000 (ver tabla 22); al mismo tiempo que se fue cambiando la razón $p = \frac{w}{W_s} = \frac{h}{H_s}$ desde $\frac{1}{32}$ hasta $\frac{1}{2}$, mientras que el valor del parámetro T_E se mantuvo en 0.1 y se utilizaron 30 imágenes par el entrenamiento y 20 para el reconocimiento

En la tabla 22 se muestran el número de errores durante los experimentos preliminares donde se varió el número de neuronas y la razón p.

Tabla 22: Número de errores en los experimentos preliminares.

Número de neuronas en la capa A	Número de errores				
	p = 1/32	p = 1/16	p = 1/8	p = 1/4	p = 1/2
2000	186	117	105	138	171
8000	138	78	66	105	132
16000	87	51	39	78	96
32000	93	42	33	66	81
64000	69	33	27	48	57
128000	57	27	18	42	51
256000	51	18	9	36	45

En este trabajo no se utilizaron distorsiones de imagen ni en las sesiones de entrenamiento ni en las de reconocimiento. La tabla 22 muestra que el número de errores de reconocimiento se reduce con el aumento del número de neuronas en la capa (A). Sin embargo, la desventaja de hacer esto radica en que a mayor número de neuronas mayor es el tiempo de entrenamiento y reconocimiento, ademas se requiere una mayor capacidad de memoria.

En los experimentos preliminares, se utilizaron 2 conexiones positivas y 3 negativas para cada neurona de la capa A. El número de neuronas de la capa A fue de 32000. Los parámetros de las ventanas fueron w = 30 y h = 30 y el tamaño de la retina fue 256×256. El número máximo de ciclos de entrenamiento fue 100.

Los siguientes experimentos se realizaron cambiando el número de imágenes, es decir, tomando un número diferente de imágenes para el entrenamiento. En el primer caso se tomaron 10/50 imágenes (Número de imágenes para entrenamiento / Número total de imágenes), después se tomaron 25/50 y finalmente 40/50. En todas estas pruebas la cantidad máxima de ciclos de entrenamiento fue 100, la cantidad de neuronas en la capa (A) fue de 32000 y el tamaño de la ventana varió de acuerdo a la razón p. Los resultados obtenidos en cada prueba se muestran a continuación en las tablas 23,24 y 25

Tabla 23: Errores obtenidos para diferentes tamaños de ventana (10/50)

Razón p	Error	% de Error
1/32	140	3.5
1/16	68	1.7
1/8	56	1.4
1/4	112	2.8
1/2	128	3.2

Tabla 24: Errores obtenidos para diferentes tamaños de ventana (25/50)

Ventana	Error	% de Error
1/32	80	3.2
1/16	32.5	1.3
1/8	30	1.2
1/4	60	2.4
1/2	72.5	2.9

Tabla 25: Errores obtenidos para diferentes tamaños de ventana (40/50)

Ventana	Error	% de Error
1/32	26	2.6
1/16	14	1.4
1/8	11	1.1
1/4	22	2.2
1/2	27	2.7

Los mejores resultados, es decir, el menor número errores, se obtuvieron con una razón de ventana ente 1/16 y 1/8 y una proporción de imágenes (40/50). Por lo tanto, estos parámetros fueron utilizados para los experimentos finales, donde se varió el número de neuronas asociativas de 32000 a 25600.

Tabla 26: Tasas de reconocimiento del clasificador en los experimentos preliminares.

Número de neuronas asociativas	Número promedio de errores de una corrida / número de imágenes en el conjunto de validación	Tasa de error (%)	Tasa de reconocimiento (%)
32000	0.44/40	1.1	98.9
64000	0.36/40	0.9	99.1
128000	0.28/40	0.7	99.3
256000	0.16/40	0.4	99.6

Los resultados de los experimentos finales se presentan en la tabla 26, donde se observa que la tasa de reconocimiento mejora al aumentar en número de neuronas en la capa (A), No obstante, para la implantación de la red en el segmento M.1 se toma como base la cantidad de 32000 neuronas ya que así se tiene un equilibrio en cuanto a recursos y se cumplen con los requerimientos del sistema, así mismo, el tamaño de ventana de escaneo es de 32×32 píxeles.

11

ARREGLO: M.1

En este capítulo se realiza el diseño de detalle del arreglo M.1, el cual es un diseño para la implementación de la red neuronal LIRA en FPGA. Se utilizaron como guía los bloques definidos en el diseño a nivel de sistema y el diagrama LIRA-FPNN propuesto. También se describe el diseño de cada uno de los componentes que conforman el segmento M.1. Al final, se realiza un recursos requeridos por el sistema.

Para comenzar se hace una recapitulación de los valores de entrada y salida de cada una de las capas pertenecientes a la red neuronal LIRA:

Tabla 27: Valores de entrada y salida

Capa	Núm de neuronas	Entradas	Salidas
Sensorial (S)	256×256	Imagen en formato BMP	Valor de brillo de los pixeles en el rango $[0, 255]$. (1Byte)
Intermedia (I)	5 (por grupo)	Valor de brillo de los píxeles en el rango $[0, 255]$. (1Byte)	Activación de neuronas ON y OFF por grupo $\{0, 1\}$ (1 bit)
Asociativa (A)	32000	Valores de neuronas ON y OFF por grupo $\{0, 1\}$	Vector de características de dimensión N con valores $\{0, 1\}$
Respuesta (R)	4	Vector de características de dimensión N con valores $\{0, 1\}$	Sumatoria de productos (por neurona)

El diseño está pensado para ser un sistema modular y escalable, enfocado a las necesidades particulares de la red neuronal LIRA-escala de grises. Sin embargo, la metodología desarrollada que integra el concepto FPNA/FPNN puede ser utilizada de igual forma para implementar otros clasificadores neuronales de su tipo.

Las redes neuronales (ANN) possen estructuras altamente paralelas y altamente interconectadas, estas características hacen que su implementación sea muy desafiante. Por lo tanto, es importante analizar diferentes metodologías para elegir la más conveniente para el tipo de ANN y su aplicación.

El desarrollo del diseño LIRA-FPNN fue realizado en base al dispositivo FPGA de *Intel MAX 10* (10M50DAF672G), el cual es un FPGA de gama media que sirve como banco de pruebas para hacer un primer análisis de la cantidad de recursos necesarios para el segmento M.1. También, se utilizó como herramienta de software *Quartus Prime Lite Edición 9.1* para el análisis y síntesis del diseño.

11.1 LIRA-FPNN

Los pasos que debe seguir el segmento M.1 para reconocer las diferentes clases de texturas del terreno marciano son los siguientes:

i La imagen es ingresada al sistema en tiempo real o desde algún dispositivo externo donde se encuentra previamente almacenada.

ii La imagen debe entrar al sistema con un tamaño de $H \times W$ píxeles, en escala de grises (1 byte por pixel) y es almacenada dentro del sistema.

iii Durante el arranque del sistema los valores entrenados y los parámetros de la red comienzan a ser cargados en la memoria del sistema para ser utilizados de manera ágil.

iv Cuando el procesamiento inicia, los datos de la imagen ingresan a los bloques de cálculo siguiendo el patrón de la máscara definida en el segmento N.1; lo mismo ocurre con los valores umbrales y los pesos entrenados.

v El computo resultante de cada bloque de cálculo se va almacenando de manera ordenada en la memoria interna del dispositivo.

vi Los valores de entrada en cada bloque se actualizan y se repite el paso iv un hasta que se completa el número de ciclos correspondiente

vii Mientras los bloques de cálculo actualizan sus valores, de forma paralela se realizan las sumatorias para cada una de las neuronas de salida.

viii De los valores de sumatoria finales se selecciona el mayor (ganador) y con esto se determina la clase bajo reconocimiento.

11.2 Bloque de entrada

11.2.1 *adquisición de datos*

Los datos correspondientes a los valores umbrales, la máscara y los valores entrenados (pesos) necesitan ser importados al segmento M.1 de forma eficiente, el principal inconveniente radica en la gran cantidad de datos y estos deben ser utilizados simultáneamente en muchos puntos de la red, lo que conlleva a establecer un método adecuado tanto para su almacenamiento, como para disponer de ellos de manera ágil.

Tal como se comprobó en los resultados del segmento N.1, la tasa de reconocimiento del clasificador aumenta al incrementar el número de neuronas asociativas. Sin embargo, esto también aumenta enormemente la cantidad de datos a importar hacia sistema M.1; para ilustrar esto se muestra a continuación el cálculo de la cantidad de datos necesarios para la red LIRA escala de grises entrenada en el capitulo 10 con 32,000 neuronas asociativas.

Tabla 28: Parámetros de la red

# de neuronas (A)	Neuronas ON	Neuronas OFF	# de clases	Tamaño de imagen
32,000	2	3	4	256×256

11.2.1.1 *Coordenadas de la máscara*

Uno de los resultados obtenidos en la etapa de entrenamiento es la generación de la máscara para el escaneo de la imagen, para esto se necesitan las coordenadas absolutas (X,Y) de los píxeles a analizar, que se obtuvieron de subventanas ($h \times w$), tantas como el número de neuronas asociativas en la red, y se tomaron tantos píxeles como número de neuronas ON/OFF existentes. Por lo tanto, la cantidad total de coordenadas absolutas se calcula como:

$$\begin{aligned} \text{Total}_X &= (\#\,\text{de neuronas A}) * (\#\,\text{de neuronas ON/OFF}) \\ \text{Total}_Y &= (\#\,\text{de neuronas A}) * (\#\,\text{de neuronas ON/OFF}) \\ \text{Total}_{XY} &= \text{Total}_X + \text{Total}_Y \end{aligned} \tag{11.13}$$

Para nuestro caso particular queda como:

$$\begin{aligned} \text{Total}_X &= (32000) * (5) = 160000 \\ \text{Total}_Y &= (32000) * (5) = 160000 \\ \text{Total}_{XY} &= 320000\,[\text{datos}] \end{aligned}$$

11.2.1.2 *Valores umbral*

Cada uno de las neuronas ON/OFF poseé valores umbrales determinados de manera aleatoria, estos valores deben ser importados al dispositivo FPGA en el mismo orden para que el sistema funcione adecuadamente. Existen tantos valores umbrales como neuronas ON/OFF en la capa I, estas neuronas se asocian en tantos grupos como neuronas asociativas existentes. Por lo tanto, la cantidad de valores umbrales se calcula como:

$$\text{Total}_{th} = (\#\,\text{de neuronas A}) * (\#\,\text{de neuronas ON/OFF}) \tag{11.14}$$

Para nuestro caso particular queda como:

$$\text{Total}_{th} = 32000 * 5 = 160000\,[\text{datos}]$$

11.2.1.3 *Pesos sinápticos*

Los pesos sinápticos son valores entrenables, se obtuvieron mediante iteraciones durante el proceso de entrenamiento de la red, cada neurona asociativa tiene conexiones ponderadas con las neuronas de la capa R (clases a reconocer). Por lo tanto, la cantidad de pesos se calcula como:

$$\text{Total}_W = (\#\,\text{de neurona A}) * (\#\,\text{de clases}) \tag{11.15}$$

Para nuestro caso particular queda como:

$$Total_W = 32000 * 4 = 128000\,[datos]$$

La tabla 29 muestra la capacidad de memoria necesaria para almacenar cada uno de los tres tipos de datos a importar. Cabe resaltar que para una cantidad de 32000 neuronas asociativas la memoria necesaria se encuentra en el nivel de Megabytes, si se quisiera aumentar el porcentaje de reconocimiento de la red y se incrementara el número a 256000 neuronas asociativas, la memoria necesaria aumentaría por un factor de 8.

Asi mismo, es necesario almacenar los píxeles de la imagen ya que estos serán tomas de acuerdo a la máscara. Para esta aplicación, la imagen de entrada se almacena en una memoria con $H \times W$ direcciones con palabras de 8 bits:

$$Capacidad = 256 \times 256 = 65536\,[palabras]$$

Tabla 29: Capacidad de memoria necesaria

Datos	Profundidad	Ancho [bits]	Capacidad de memoria [MB]
Coordenadas	320,000	8	2.56
Umbrales	160,000	8	1.28
Pesos	128,000	8	1.024
Imagen	65536	8	0.525

Se pueden elegir diferentes tecnologías de memoria para almacenar los datos anteriores, el objetivo es acceder a ellos de manera rápida y eficiente, para ello cada tecnología ofrece diferentes ventajas que deben tomarse en cuenta según las necesidades; por ejemplo, las memorias DRAM (RAM dinámica) muy compactas, sin embargo, no pueden ser sintetizadas y se requiere un chip por separado, por otro lado las memorias SRAM son mucho más rápidas y pueden ser sintetizadas en un FPGA, por lo que son ideales para registros o bloques de memoria pequeños y rápidos. También, cabe mencionar que la memorias FLASH son fáciles de usar y se encuentran actualmente disponibles en muchas tarjetas de desarrollo FPGA. Para este trabajo, se considera que los datos se encuentran previamente cargados en memorias SRAM las cuales son llenadas en un tiempo anterior al arranque del sistema.

11.2.2 *Generador de números pseudoaleatorios*

Existe otra opción para importar los valores de la red neuronal LIRA en el FPGA, y es generarlos en tiempo real (excepto los pesos sintéticos), para ello es necesario contar con generadores de números pseudoaleatorios que generen la misma cadena de valores utilizados en la etapa de entrenamiento. Como se describe E, el generador lineal congruencial (GLC) es una opción muy eficiente para ser implementada en FPGA ya que solo utiliza un multiplicador, un sumador y la operación módulo se realiza mediante el truncamiento de las palabras resultantes. Además, se puede hacer más eficiente si se utiliza los multiplicadores embebidos que poseen

los FPGA actuales como macros de hardware.

EL número máximo de GLCs que se necesita para generar los valores en tiempo real es cinco como se muestra a continuación:

- 2 Generadores para las coordenada absoluta X
- 2 generadores para la coordenada absoluta Y
- 1 generador para los valores umbrales

En la Figura 84 se muestra un diagrama esquemático del GLC implementado en FPGA, utiliza los parámetros a,c y M para BorlandC/C++ (ver tabla 34). Por lo tanto, resulta contener un multiplicador de 32 bits, el cual para ser implementado en el FPGA *MAX 10* se utilizan tres multiplicadores embebidos como macros de hardware.

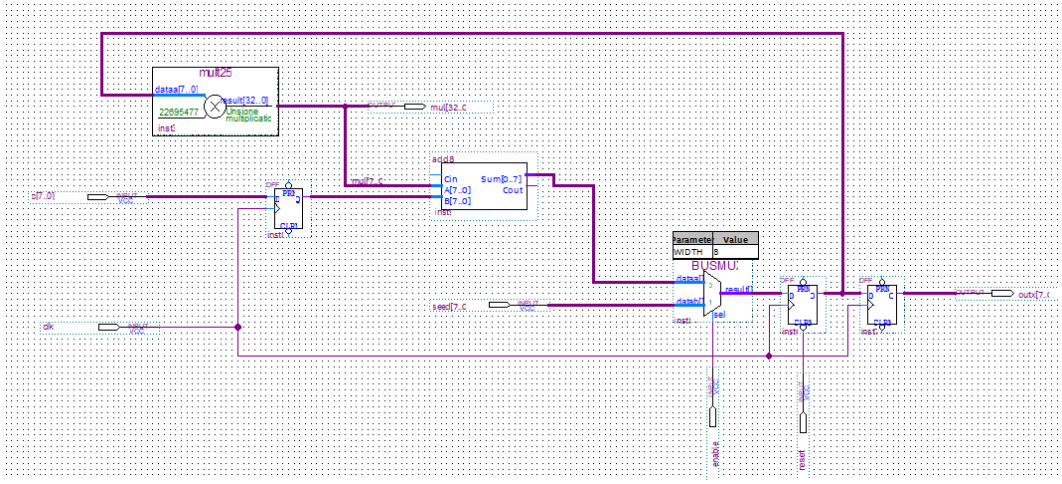


Figura 84: Diagrama esquemático del GLC en FPGA

11.2.3 *Memoria del sistema*

Los valores neuronales importados al dispositivo, así como los datos de entrada, deben almacenarse temporalmente en la memoria del sistema para acceder a ellos de manera rápida y eficiente. Uno de los mejores enfoques para proporcionar memorias funcionales y eficientes en los dispositivos FPGA, es el introducir bloques de memoria embebidos como macros de hardware (BRAM), estos módulos de memoria son denominados macroceldas y están separadas de las celdas lógicas normales; los dispositivos FPGA contienen o no un determinado número de BRAM, dependiendo de la marca y la familia del dispositivo, con capacidades expresadas [Mb].

Para la red LIRA-FPNN, los valores de entrada son almacenados en de bancos de memoria, los cuales se construyen mediante tres memorias BRAM, una para cada tipo de valor, configuradas cómo memorias FIFO (First in-First out), tal como se muestra en la figura 85.

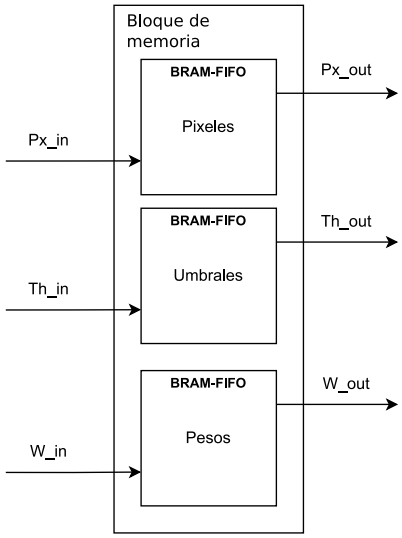


Figura 85: Bloque de memoria con tres memorias BRAM

Estos bloques de memoria se repiten en el diseño tantas veces como columnas existentes en la matriz FPNN, í.e. para una matriz de N × M bloques de cálculo, se colocan M bloques de memoria. En este trabajo se utilizó una matriz de treinta y dos bloques de cálculo (8 × 4), por lo tanto se tienen cuatro bloques de memoria en el diseño (ver fig. 86).

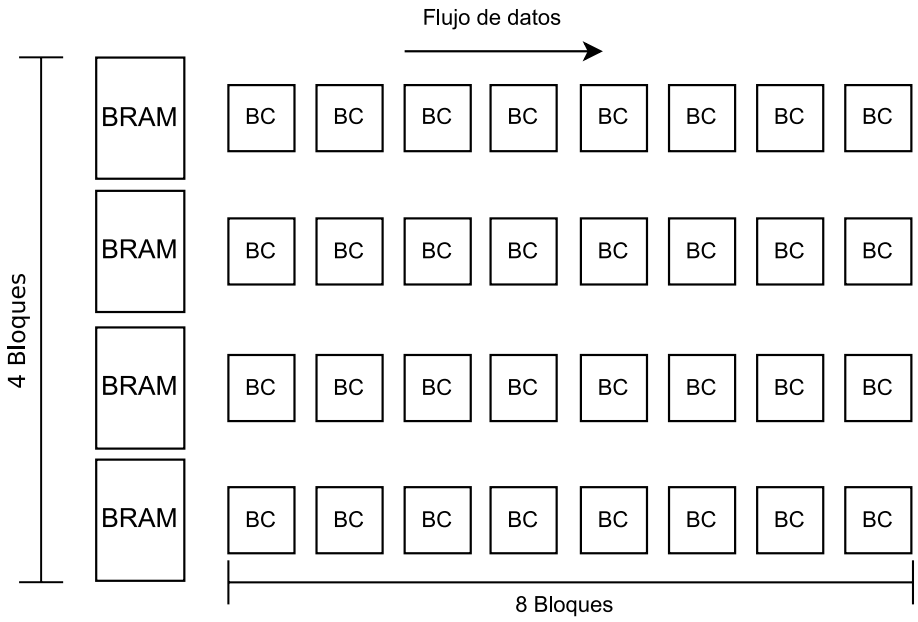


Figura 86: Matriz con bloques de calculo y cuatro bloques de memorias

El diseño de la memoria FIFO-BRAM, implica principalmente el cálculo del su capacidad (profundidad * ancho), para esto es necesario conocer la cantidad de datos necesarios a almacenar, la velocidad a la que son escritos y la velocidad a la que son leídos.

De forma estadística, la velocidad de transmisión de datos varía en el sistema, depende principalmente de la carga existente. Por lo tanto, para obtener un tamaño

de memoria seguro se considera el peor escenario para la transferencia de datos. El modelo matemático utilizado se muestra a continuación.

11.2.3.1 *Modelo matemático para FIFO-BRAM:*

- Frecuencia de escritura - F_w
- Frecuencia de lectura- F_r
- Longitud de ráfaga[1] - B
- Número de ciclos inactivos entre dos escrituras sucesivas - $idle_w$
- Número de ciclos inactivos entre dos lecturas sucesivas - $idle_r$

Por lo tanto, la profundidad de la memoria FIFO se calcula como:

$$FIFO_{depth} = B - \frac{B * F_r}{F_w * idle_w} \tag{11.16}$$

Lo más importante en la estimación del tamaño de la memoria FIFO, es evitar la perdida de información, en el caso particular en que la frecuencia de lectura es mayor a la frecuencia de escritura, una memoria con profundidad igual a uno bastaría, en la arquitectura de LIRA-FPNN pasa algo similar, los datos se escriben en los cuatro bloques de memoria (ver fig. 86) de manera secuencial del primero al último, sin embargo cada bloque es leído de manera paralela en cada fila de la matriz, lo que incrementa la velocidad de lectura con respecto a la escritura de los bloques.

Por lo tanto, se determina una profundidad de memoria FIFO igual a la longitud de ráfaga y una frecuencia de escritura de por lo menos tres veces la frecuencia de lectura, con el fin de evitar tiempos muertos.

11.2.3.2 *Descripción de las memoria BRAM*

Existen varios métodos para incorporar módulos BRAM en un diseño, los métodos más comunes son los siguientes:

- Mediante instanciación de HDL
- Mediante los programas de utilidad (*Block Memory Generator* y *Distributed Memory Generator*)
- Mediante una plantilla de inferencia de HDL conductual

Los dos primeros métodos son métodos específicos para ciertos dispositivos Xilinx, sin embargo pueden llegar a ser tediosos y generan código difícil de entender. No obstante, el tercer enfoque consiste en utilizar plantillas HDL conductuales para inferir el módulo de memoria interna.

1 Número de elementos a transferir

Los fabricantes comúnmente, proporcionan plantillas sugeridas de HDL para configurar dichas memorias, las cuales conviene seguir, ya que de lo contrario podrían surgir problemas con la síntesis del módulo y la memoria descrita se configuraría mediante los bloques lógicos del dispositivo en lugar de utilizar las macroceldas. Dicho lo anterior, para el diseño de LIRA-FPNN se utilizó el tercer enfoque, la inferencia de HDL conductual mediante una plantilla.

Utilizando la plantilla se describió cada una de las memoria BRAM con los parámetros requeridos y con configuración FIFO. También, se incluyó un bloque de control para cada memoria (ver fig. 87), con el cual es posible sincronizar la lectura de datos y habilitar la escritura, también incluye las señales *full* y *empty*, las cuales indican cuando la memoria se encuentra llena y vacía respectivamente.

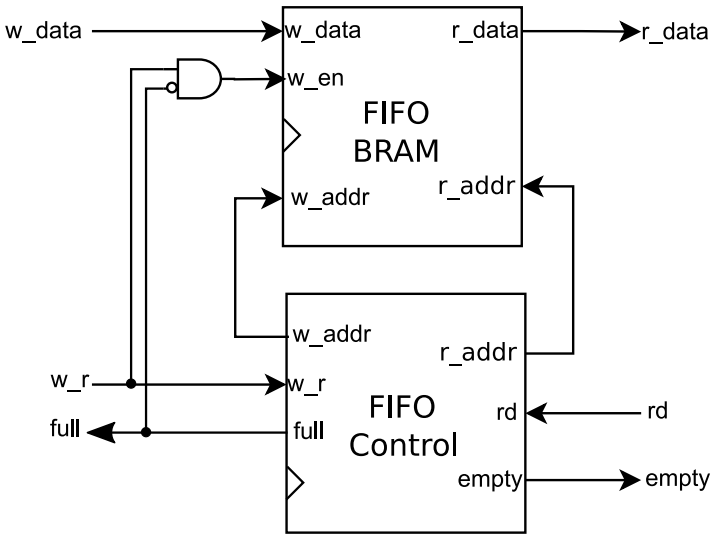


Figura 87: Diagrama de bloques de una memoria FIFO basada en BRAM

En la figura 88 se muestra la representación esquemática de la descripción de un bloque de memoria, contiene una señal adicional (clr) la cual sirve para limpiar las memorias al terminar el procesamiento de cada imagen y dejarlas listas para la siguiente imagen a reconocer.

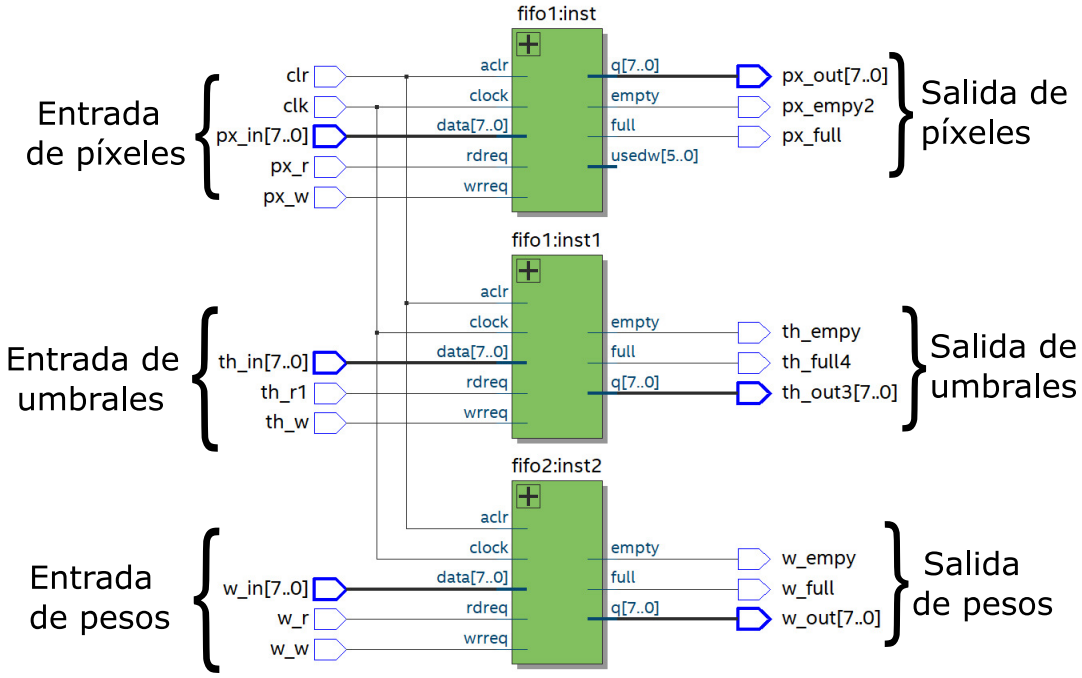


Figura 88: Bloque de memoria con tres FIFO

11.3 Bloque de cálculo

El bloque de cálculo opera sobre todos los datos de entrada del sistema, realiza el cálculo correspondiente a las capas intermedias de la red (capa I, capa A) así como el producto del vector binario por los pesos sinápticos ($\vec{N} \times \vec{W}$).

En la figura 89 se muestra un diagrama simple de los elementos que conforman el bloque de cálculo; contiene dos bloques: *Coding* y *Synapse*, el primero realiza las operaciones de las capas intermedias (I, A), mientras que el segundo habilita la utilización de los valores de peso para la entrada de datos en turno.

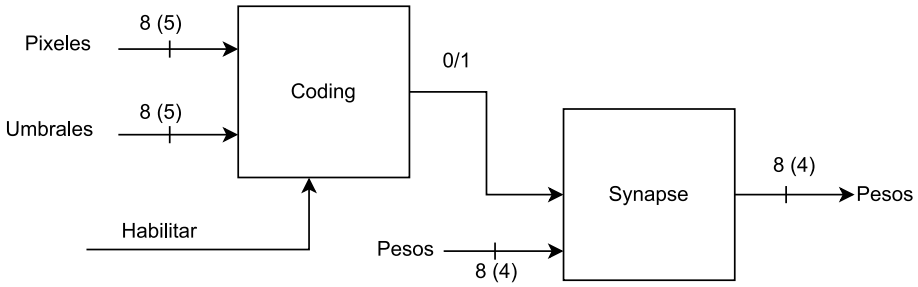


Figura 89: Diagrama del bloque de cálculo

Para el caso particular de la red neuronal LIRA, no se necesitan realizar operaciones de multiplicación con los pesos sinápticos como tal, ya que al ser un vector binario sus elementos serán cero o uno, y el resultado de dicha multiplicación sera a su vez cero o el valor de peso en operación.

A continuación se muestra en la figura 90 los elementos que conforman el bloque de cálculo descritos en el dispositivo FPGA, como se puede observar el bloque

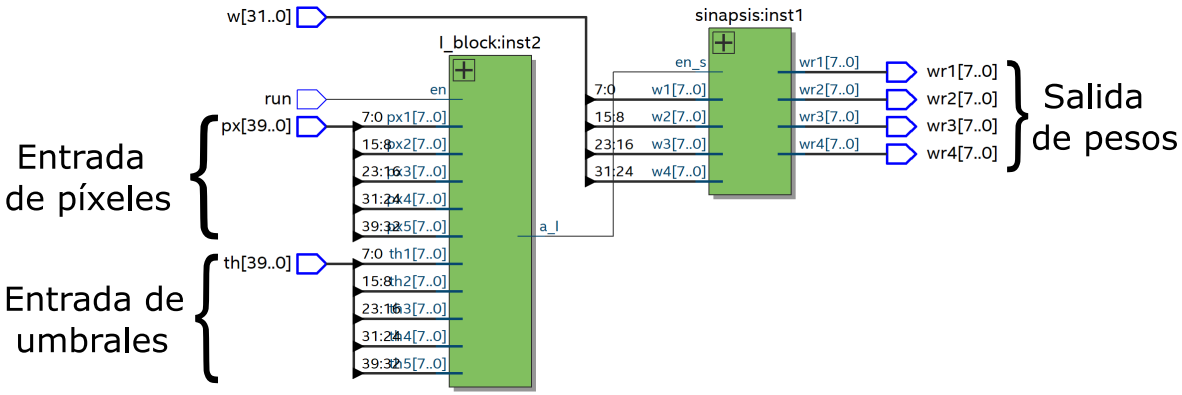


Figura 90: Vista RTL del bloque de cálculo

synapse es habilitado por el bloque anterior *Coding* mediante las señal a_l, la cual corresponde a la salida de la neurona asociativa en turno [0,1]. La señal debe estar en un estado alto (1) o de lo contrario el bloque Sinapsis no se habilitará y dejará pasar el valor de peso correspondiente, en su lugar dara a la salida cero.

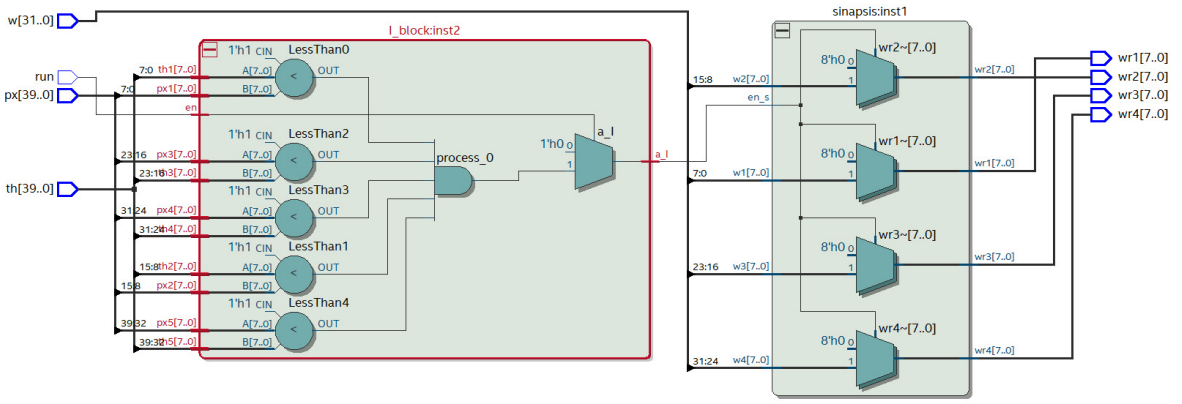


Figura 91: Vista RTL expandida del bloque de cálculo

En la figura 91 puede apreciarse una vista expandida del bloque del cálculo utilizando la herramienta de visor RTL que forma parte de la suite de diseño de Quartus Prime; las siglas RTL significan lógica de transferencia de registro o nivel de transferencia de registro, RTL se basa en el concepto de que el movimiento de los datos a través de la lógica digital puede verse como diagramas lógicos con expresiones que representan a las variables de datos almacenadas en los registros, generalmente es un nivel de representación más alto que el nivel de compuerta.

En el bloque del lado izquierdo de la figura 91 (*Coding*) los valores de los píxeles de la imagen se comparan con los valores umbrales de las neuronas ON-OFF, si se cumplen todas las condiciones la salida de este bloque será uno, lo que habilitará al bloque del lado derecho (Synapse), y este a su vez dejara de pasar los valores de peso para los cálculos posteriores.

11.3.1 *Entrada de datos*

Cada uno de los tres tipos de datos almacenados en los bancos de memoria son transferidos a la entrada de los bloques de cálculo mediante registros de desplazamiento. en la figura 92 se muestra el diagrama de un registro de desplazamiento para la carga de cinco valores hacia un bloque de cálculo, cabe aclarar que los flip flops en la figura no representan flip flop individuales, sino que representan registros de ocho bits.

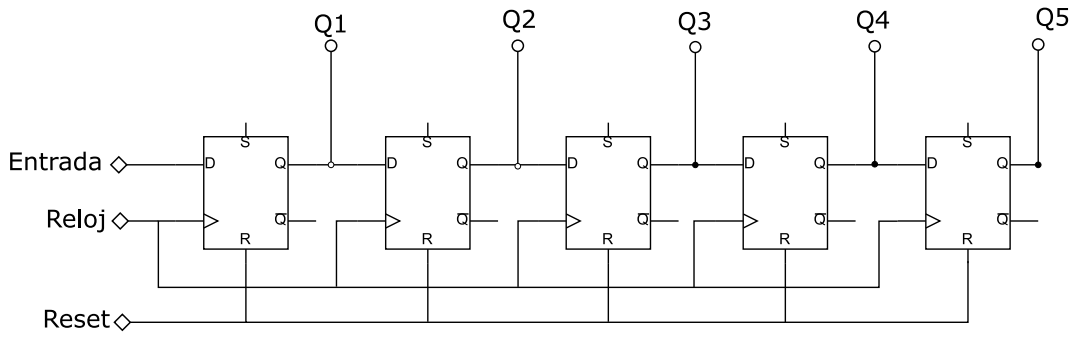


Figura 92: Registro de desplazamiento para 5 valores

También, en la figura 93 se muestra la representación esquemática del registro de desplazamiento mediante el visualizador RTL. Para este trabajo, se tienen tres registros para cada bloque cálculo, uno para cada tipo de dato, dos de los registros cargan cinco bytes en paralelo (umbrales y píxeles) y el tercero carga cuatro bytes (pesos).

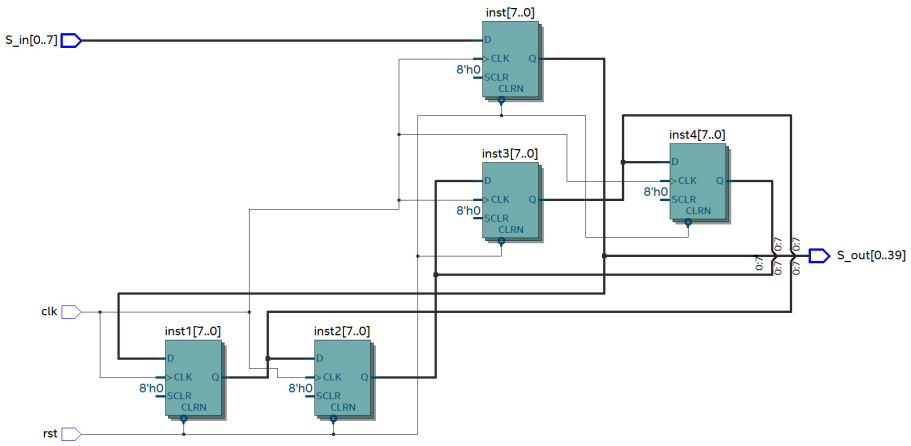


Figura 93: Registro de desplazamiento en vista RTL

Cada registro de desplazamiento de cada bloque de cálculo se interconecta entre sí con sus vecinos de manera que se forma un gran registro de desplazamiento para cada fila de la matriz LIRA-FPNN; en la figura 94 se muestran seis registros de cinco bytes conectados entre si para desplazar los datos desde extrema izquierda hasta extrema derecha.

En el diseño de LIRA-FPNN se utilizan treinta y dos bloques de calculo, ocho en cada fila, por lo que se tiene en cada fila tres grandes registros de desplazamiento

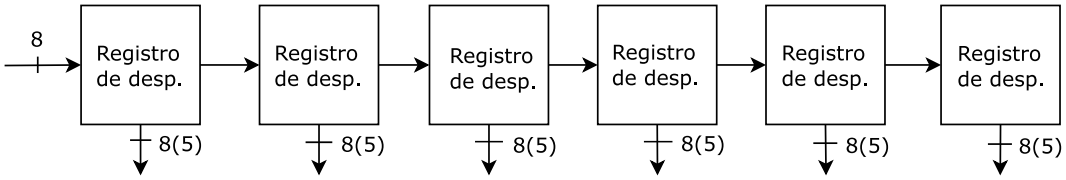


Figura 94: Diagrama de registros de desplazamiento conectados

creados cada uno con ocho registros mas pequeños; la idea de describir unidades lógicas pequeñas y configurarlas para formar otras más grandes se hizo con la idea de que sea un sistema escalable. En la figura 95 se muestra la representación esquemática de una fila de ocho bloques de cálculo con sus registros de desplazamiento para ingresar los datos.

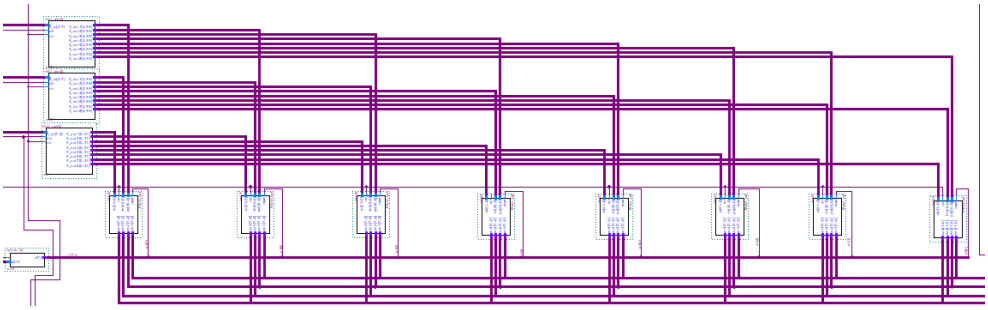


Figura 95: Fila de LIRA-FPNN con registros de desplazamiento

11.4 Bloque de salida

Los datos proporcionados por cada bloque de cálculo deben ser separados y ordenados para su procesamiento final, para lograr esto se utilizan nuevamente memorias BRAM-FIFO como elementos de almacenamiento en cola; ahora se tienen tantas memorias como número de clases a reconocer, o dicho de otra forma, tantas memorias como el número de neuronas en la capa (R).

Todos los valores de salida de los bloques de cálculo que corresponden a una neurona (R) comparten un mismo bus de datos, es decir, existe un bus por cada clase a reconocer; este bus se implementó a través de buffer tri estado y demultiplexores para seleccionar la lectura de cada bloque por fila y por columna.

Estas memorias se irán llenando esencialmente con sus valores correspondientes de los bloques de cálculo, el muestreo de datos se realiza en la matriz de bloques de izquierda a derecha y de arriba a abajo. Utilizando la ecuación 11.16 y se estima una capacidad para los cuatro FIFO de 32 bits de profundidad; en a figura 96 se muestra la vista RTL de los elementos de selección para el ordenamiento de datos.

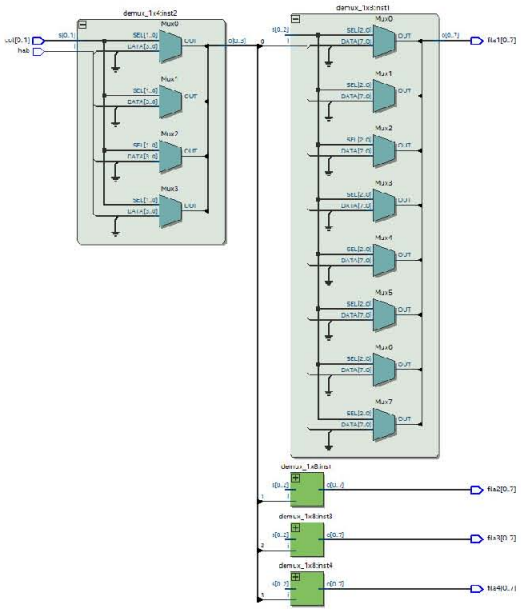


Figura 96: Vista RTL de los demultiplexores para la selección

11.4.1 *Sumador*

Un sumador completo se puede construir con tan solo cinco compuertas como se muestra en la figura 97, en los FPGA actuales se utiliza una LUT de tres entradas para implementarlo.

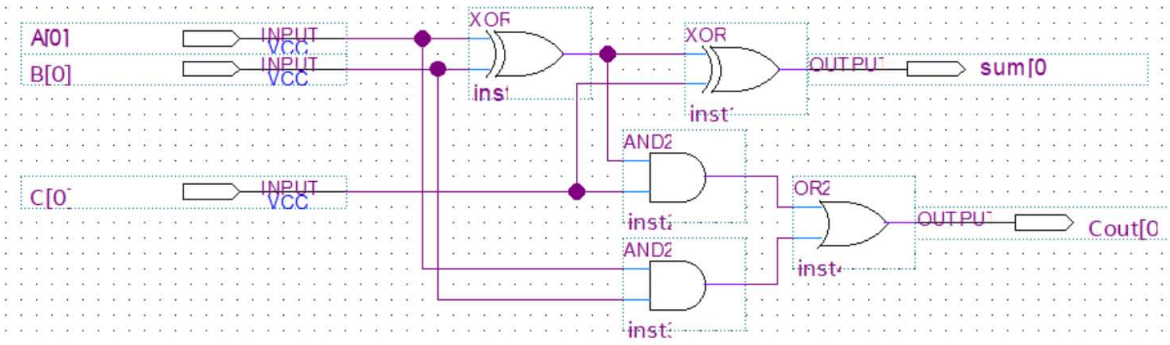
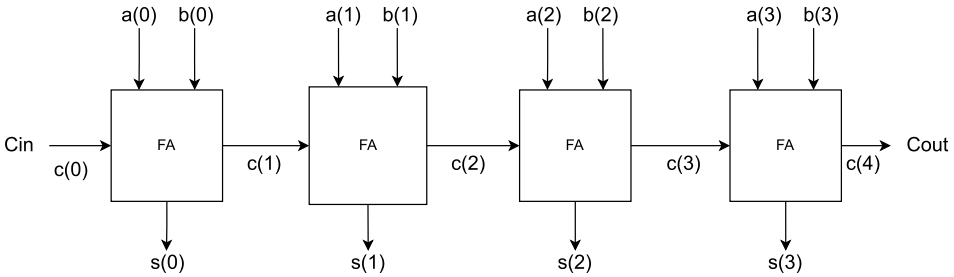


Figura 97: Sumador completo

EL sumador con propagación de acarreo (RCA, por sus siglas en inglés) y el sumador con anticipación de acarreo (CLA, por sus siglas en inglés) son dos enfoques clásicos para el diseño de sumadores. El primero tiene la ventaja de requerir menos hardware, mientras que el segundo es más rápido. Ambos enfoques se analizan a continuación.

El sumador RCA mostrado en la figura 98 es fácil de diseñar, sin embargo es relativamente lento en cuanto a rendimiento, ya que cada sumador completo debe esperar al bit de acarreo calculado en el sumador completo anterior.

El retraso total a través del sumador RCA para la suma de salida se expresa por la ecuación:



a	b	cin	s	cout
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

Figura 98: Sumador con propagación de acarreo

$$T_{sum}(n) = (n-1) * T_c + T_s = (n-1) * 3D + 2D = (3n-1)D \tag{11.17}$$

Donde D es el retraso a través de una compuerta, y n es el número de bits en el sumador. Por lo tanto, utilizando la ecuación 11.17 se sabe que para un sumador RCA de 24 bits se tendrá 71 retrasos de compuerta.

Por otro lado el sumador CLA mostrado en la figura 99 esta creado con circuitos más sofisticados con los cuales se puede mejorar el rendimiento.

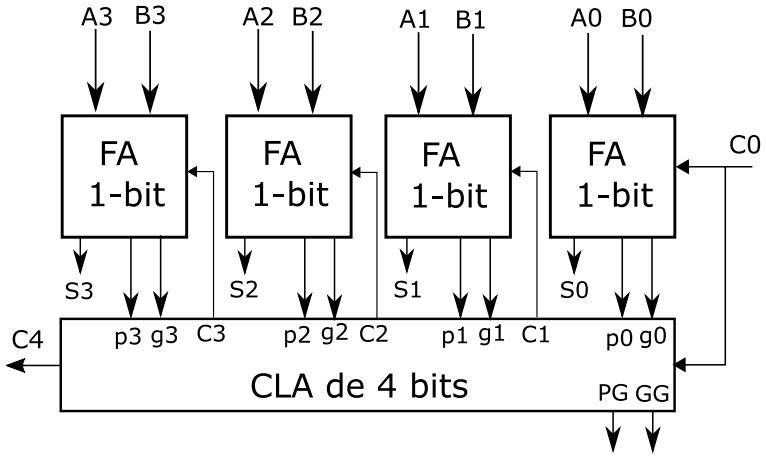


Figura 99: Sumador con anticipación de acarreo

En el sumador CLA, el bit de acarreo se puede calcular como términos de generación y propagación, por ejemplo, si se tiene:

$$C_i + 1 = G_i + P_i.C_i$$

en donde

$$G_i = A_i.B_i$$

y la propagación P:

$$P_i = A \text{ XOR } B$$

Se tiene ahora que el retraso en el sumador CLA de n bits es aproximadamente igual a n, lo que es una gran mejora de rendimiento sobre el sumador RCA.

En la figura 100 se muestra a través de la herramienta *Technology Map Viewer* el resultado de la implementación de un sumador RCA de 4 bits en un FPGA Altera MAX 10; muestra que la implementación está hecha de una cascada de cuatro pares de LUT de tres entradas, el retraso de este circuito será entonces de solo cuatro retrasos de LUT y no de once retrasos de compuerta como lo indicaría la ecuación 11.17. Además, puede mejorar aún más si se utilizan arquitecturas FPGA más avanzadas.

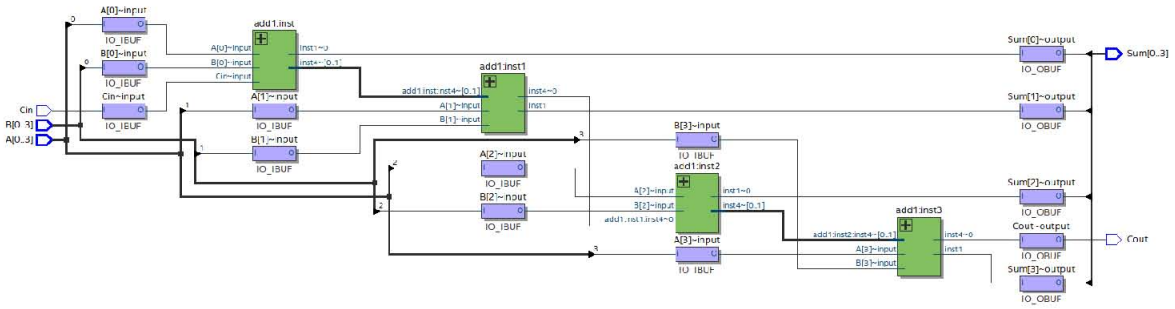


Figura 100: Vista del sumador completo de 4-bit

Aunque el uso del CLA u otros circuitos sumadores más sofisticados puedan ser más beneficiosos en algunas aplicaciones, la presencia de arquitecturas FPGA de grano grueso puede hacer esto innecesario en muchos casos; la implementación elegida debe determinarse caso por caso ya que hay muchas compensaciones que pueden no ser obvias al principio. El uso de LUT y circuitos aritméticos adicionales como cadenas de acarreo y sumadores completos hacen que los FPGA implementen sumadores particularmente eficientes.

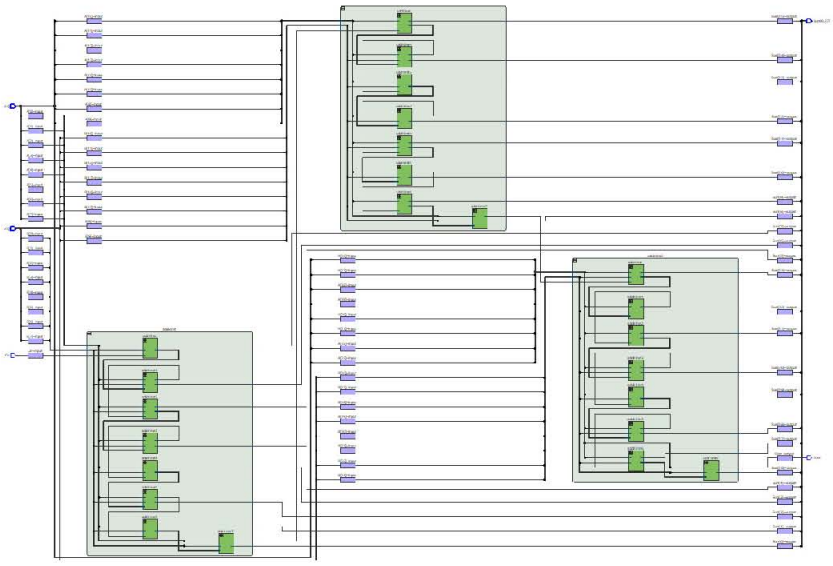


Figura 101: Vista del sumador completo de 24-bit

Los resultados de los bloques de cálculo organizados en las cuatro memorias FIFO son utilizados para realizar las sumatorias totales de cada una de las neuronas de salida; el resultado de cada suma se almacena en un acumulador el cual se suma al dato siguiente hasta completar la sumatoria total. Aunque los datos almacenados en las memorias FIFO son de 8 bits, el hecho de ser una gran cantidad datos a

sumar genera un desbordamiento en el tamaño de palabra del acumulador. Por lo tanto es necesario utilizar sumadores de 24 bits tal como se muestra en la figura 101.

11.4.2 *Selección de ganador*

El bloque final de LIRA-FPNN es el selector de ganador, el cual se encarga de determinar cual de las cuatro sumatorias totales (salidas de las neuronas R) tiene mayor valor; cabe mencionar que a diferencia de la etapa de entrenamiento de la red donde el mayor valor se comparaba con un valor competidor y el ganador era modificado o no, aquí al tratarse unicamente de procesos de reconocimiento no es necesario hacer esto y simplemente se define como ganador a la salida con el valor más grande.

En la figura 102 se muestra una vista expandida, utilizando la herramienta RTL, del selector del ganador. Los resultados de las sumatorias ingresan al selector y son comparadas entre si, el resultado da mediante una palabra de tres bits, donde la interpretación es la siguiente:

Tabla 30: Interpretación de la salida de LIRA-FPNN

Salida	Clase
001	Arena
010	Cielo
011	Parche
100	Roca

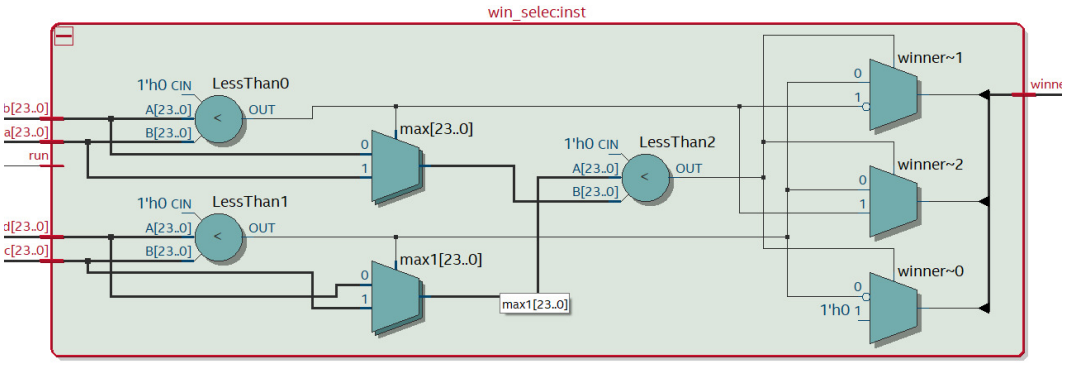


Figura 102: Vista RTL del selector de ganador

En a figura 103 se muestra los elementos del bloque de salida para LIRA-FPNN, de izquierda a derecha se encuentran las memorias FIFO, maquinas para completar las palabras a 24 bits, sumadores completos, acumuladores y por último el selector de ganador.

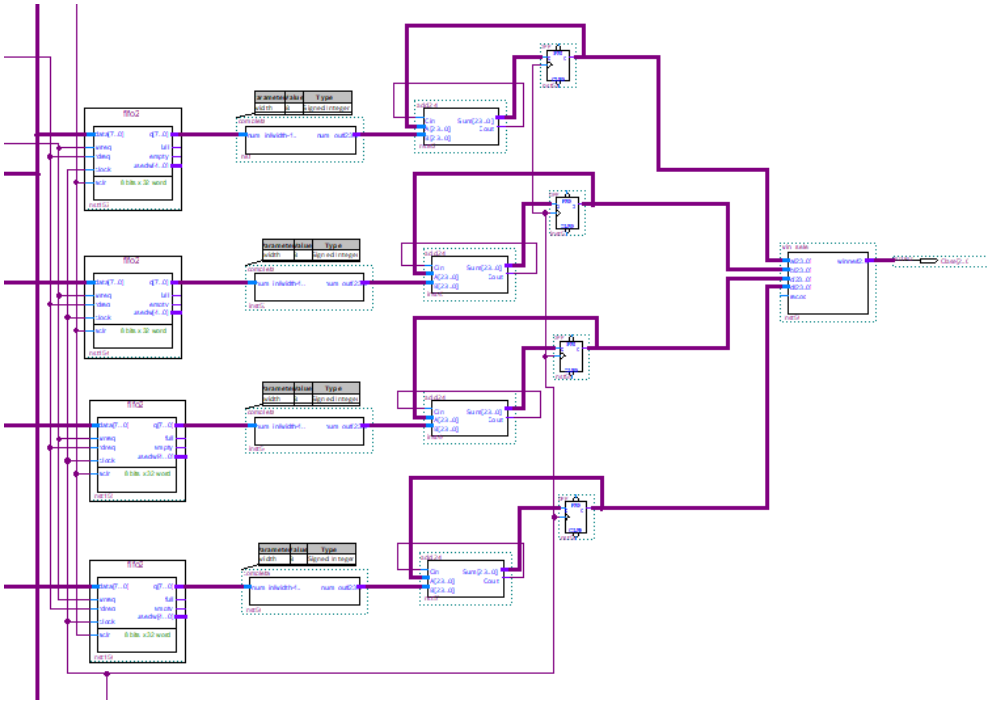


Figura 103: Elementos del bloque de salida

11.5 Análisis de diseño

11.5.1 *Análisis de recursos*

Después de la descripción y la compilación del diseño en el FPGA seleccionado, fue posible conocer la cantidad de recursos necesarios para su implementación. A través de la herramienta de software se realizaron dos tipos de compilación para el diseño, con el fin de comparar su desempeño; la primera fue una compilación balanceada (normal), mientras que la segunda fue una compilación agresiva, enfocada a la optimización de tiempo y área.

El *planificador de chip* (chip planner, en inglés) permite un despliegue visual de los recursos del dispositivo y como estos se encuentran distribuidos, el planificador de chip es de gran ayuda para establecer un plan de diseño y realizar modificaciones posteriores a la compilación; en este trabajo se utiliza para analizar la colocación de los recursos en el dispositivo y para buscar la causa de posibles problemas de sincronización.

En la figura 104 se muestra el planificador de chip con el diseño LIRA-FPNN, donde los arreglos de bloques lógicos (LAB, por sus siglas en inglés) se muestran de color azul y los bloques RAM se muestran como columnas verticales de color verde oliva. EL planificador de chip utiliza una combinación de colores degradados en la que el color se vuelve más oscuro a medida que aumenta la uso del recurso en especifico (ver fig. 104-derecha).

Para la estructura de LIRA-FPNN con treinta y dos bloques de cálculo, se necesitan aproximadamente 4524 elementos lógicos, 6144 bits de memoria BRAM y 3 multiplicadores embebidos de 9 bits por cada GLC que se implemente.

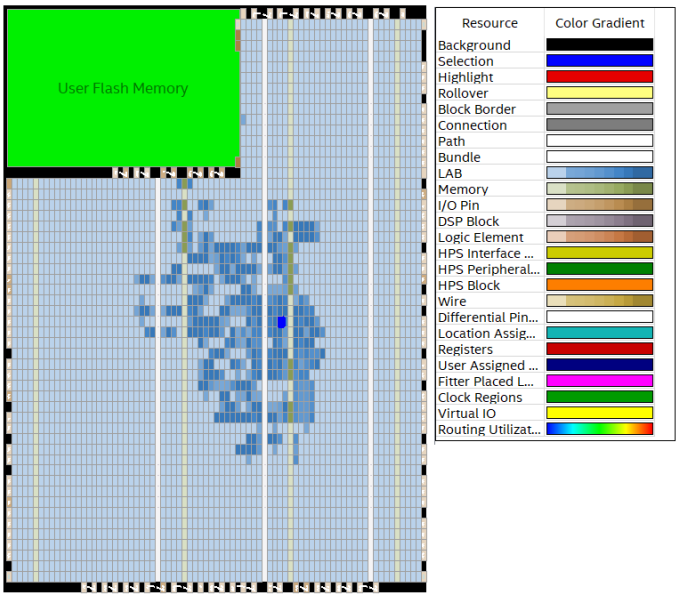


Figura 104: Vista del planificador de chip

11.5.2 *Análisis de potencia*

Una vez completado el diseño LIRA-FPNN, se realizaron estimaciones para el consumo de energía, una de ellas fue mediante la herramienta *Power Play Power Analyzer*, con la cual se obtuvo un valor estimado de 98.5 mW (ver figura 105)

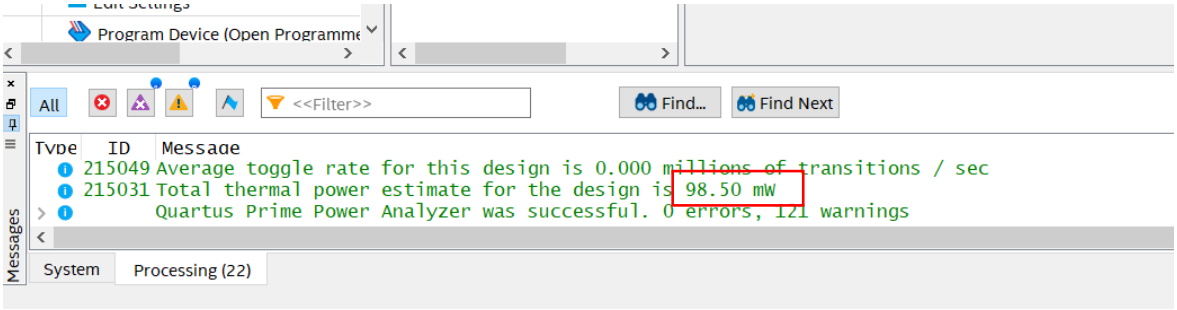
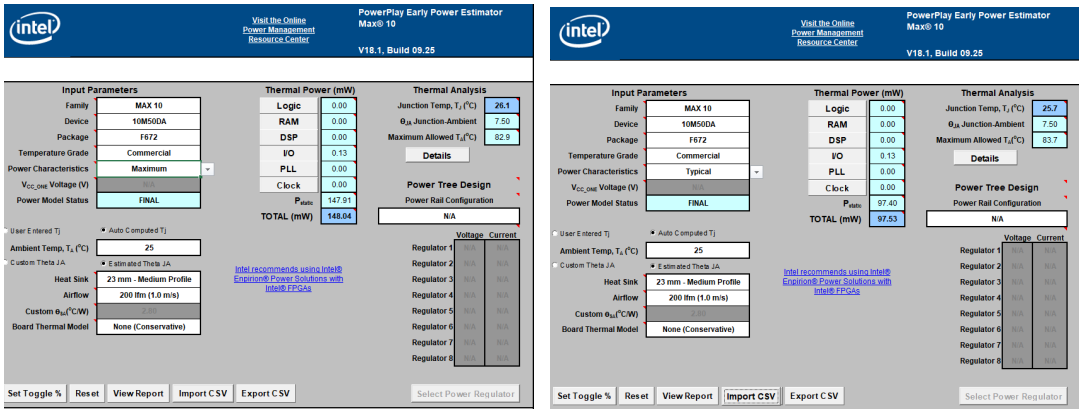


Figura 105: Potencia utilizando Power Play Power Analyzer

Otro de los métodos utilizado para estimar la potencia, fue utilizando el *estimador temprano de potencia* (Early Power Estimator) que es una herramienta de análisis mediante hojas de cálculo que permite realizar estimaciones de potencia en base a la selección del dispositivo, paquetes, utilización de recursos y condiciones operativas. Como su nombre los indica es un estimador temprano y aunque sus modelos de estimación son bastantes precisos en general, se requieren estimaciones de las velocidades de todos los relojes del dispositivo e información de enrutamiento para obtener aún mejores resultados. Por lo tanto, las estimaciones posteriores utilizando información real de enrutamiento y ubicación dentro del dispositivo será

considerablemente más exactas. No obstante, obtener la estimación temprana es de gran utilidad para el diseño de la fuente de alimentación para el dispositivo.

Los datos necesarios para el estimador temprano pueden ingresarse manualmente o importarse en formato .CSV y con ellos se puede estimar el presupuesto de energía para cada voltaje de la fuente de alimentación y permite un análisis rápido del efecto que tendrán los cambios de diseño en el consumo de energía. En la figura 106 se muestra los resultados del estimado de potencia máxima y típica para el diseño LIRA-FPNN los cuales son 148 mW y 97.53 mW respectivamente.



(a) Potencia máxima (b) Potencia Típica

Figura 106: Estimador de potencia temprano

11.5.3 *Análisis de tiempos*

EL control de los procesos LIRA-FPNN para el reconocimiento de imagen está conformado principalmente por contadores, los relojes utilizados proporcionan la sincronización necesaria para la transferencia de datos sin errores. Para evitar que el valor de los datos sean incorrectos en el tiempo en que estos hacen la transición de un valor a otro, se debe esperar a que los datos se estabilicen antes de utilizarlos, o mas específicamente, los datos deben ser estables antes y después del borde de reloj para ser transferidos de forma fiable.Para evitar problemas de sincronización es necesario determinar una frecuencia máxima de reloj con la cual no se violen los requisitos del tiempo de establecimiento (t_{SU}) y el tiempo de retención (t_H) de datos. En la tabla f_{MAX} es la frecuencia máxima a la que se puede ejecutar el diseño. Por otro lado, la frecuencia restringida f_{MAXr} es la frecuencia máxima a la que puede ejecutarse el diseño considerando los límites del dispositivo tales como las velocidades máximas de alternancia.

Tabla 31: Frecuencia máxima

Frecuencia	MHz @85°C	MHz @0°C
F_{MAX}	60.4	63.24
F_{MAXr}	64.99	69.23

Parte IV

RESULTADOS Y CONCLUSIONES

12

ANÁLISIS DE RESULTADOS

12.1 Resultados de N.1

El clasificador neuronal LIRA para el reconocimiento de texturas de superficies del terreno marciano fue probado en diferentes experimentos para determinar los valores más adecuados de sus parámetros para la aplicación.

En la figura 107 se muestran los resultados de los experimentos relacionados al aumento de neuronas asociativas en la capa (A), las diferentes barras de colores representan los experimentos con diferentes tamaños de ventana. En la gráfica se puede observar que a mayor número de neuronas asociativas disminuye el número de errores y por ende aumenta el porcentaje de reconocimiento del sistema. No obstante, aunque aumentar la cantidad de neuronas asociativas mejora el reconocimiento también incrementa la cantidad de valores a almacenar y con ello la memoria necesaria para la implementación, también podría aumentar el tiempo total de reconocimiento ya que el número de iteraciones en cada bloque de cálculo esta ligado al número de neuronas asociativas. Sin embargo, esto último se pude solucionar al aumentar el número de bloques de cálculo en la estructura FPNN por lo que en lugar de necesitar más tiempo para el reconocimiento se necesitara utilizar más recursos en el FPGA.

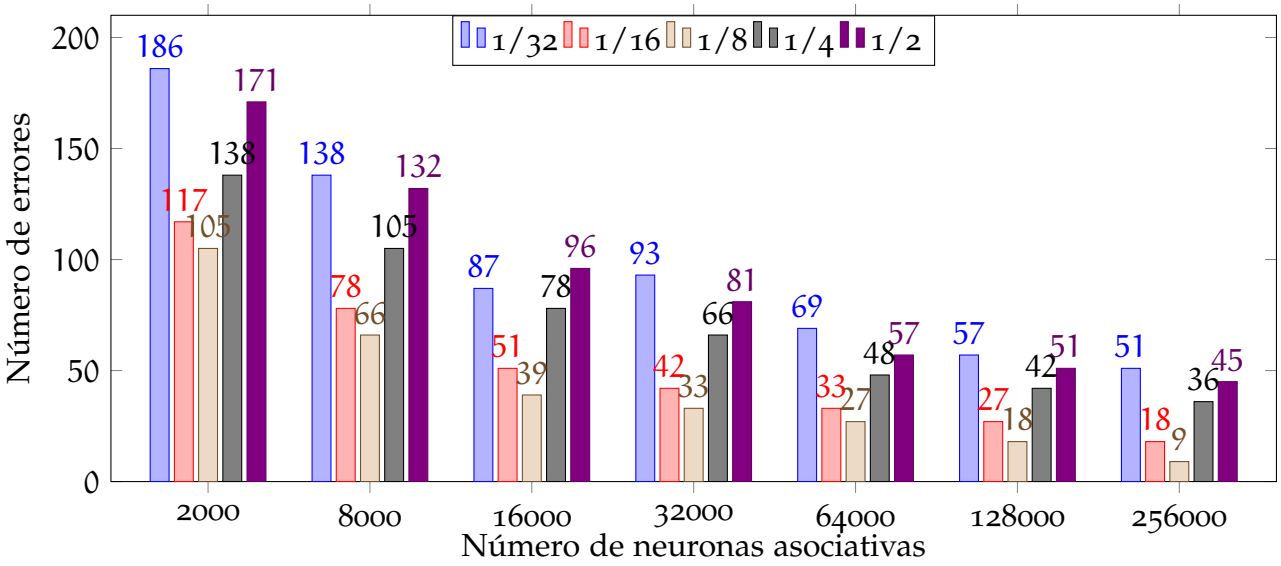


Figura 107: Número de errores vs número de neuronas

En la figura 108 se muestra la gráfica del porcentaje de error en función del tamaño de ventana de escaneo, representado mediante la razón P, las barras de colores representan cada uno de los experimentos donde se varia el número de imágenes para entrenamiento y reconocimiento. En la gráfica se puede apreciar que los mejores resultados se producen con un valor de ventana intermedio donde los errores

se minimizan, los valores muy pequeños o muy grandes no dan tan buenos resultados, esto se debe a que para una ventana relativamente pequeña las características de la imagen se presentan mejor que para una ventana mucho más grande, no obstante, si la ventana es demasiado pequeña se tiene mucho menos información para representar adecuadamente las características de la imagen. Por lo tanto, el valor de $P = 1/8$ que en nuestro caso es una ventana de 32×32 píxeles es el valor que proporciona el mejor porcentaje de reconocimiento.

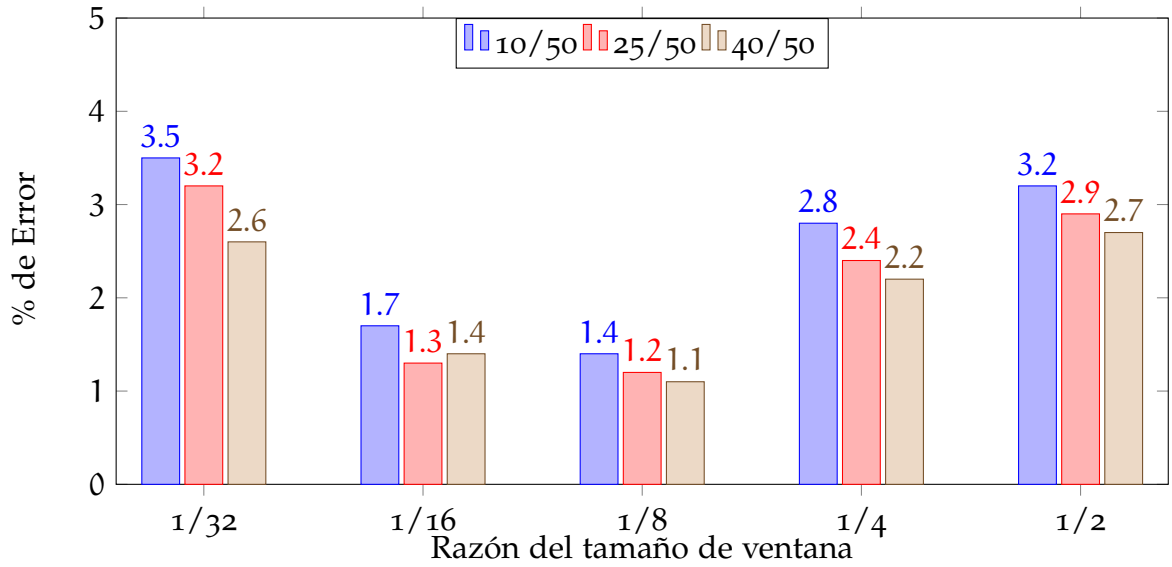


Figura 108: Porcentaje de error vs tamaño de ventana

A continuación, en la tabla 32 se presentan las especificaciones finales del arreglo N.1, el tamaño de la capa (A) de 32000 neuronas se eligió por ser un valor equilibrado con el cual se logra un porcentaje de reconocimiento aceptable sin requerir de una gran cantidad de recursos lógicos para su implementación. La base de datos ITM puede actualizarse para incluir más imágenes de cada clase o para añadir nuevas clases para el reconocimiento.

Tabla 32: Especificaciones finales de N.1

Núm	Núm. de Necesidad	Métrica	Imp.	Unid.	Valor final
1	13	Tamaño de la imagen	5	px	256 × 256
2	13	Tamaño de ventana	5	px	32 × 32
3	1,13	Profundidad de bits	4	bits	8
4	16	Porcentaje de reconocimiento	5	%	98.9
5	4,7	Capas	5	Num	4
6	4,7	Tamaño de la capa A	4	Neuronas	32000
7	13	Imágenes para entrenar	5	Num	160
8	13	Imágenes para reconocimiento	5	Num	40
9	7	Ciclos de entrenamiento	-	Num	100
10	13	Clases a reconocer	5	Num	4

12.2 Resultados M.1

Los valores entrenados, los datos de la máscara, umbrales necesitan ser importados al sistema y por lo tanto demandan una cantidad de memoria externa para ser almacenados. En la figura 109 se muestra una gráfica con el porcentaje de memoria que requiere cada tipo de dato, las coordenadas de la máscara utilizada para el escaneo de la imagen son el tipo de dato que ocupa más espacio en la memoria y además, al igual que los valores umbrales, es proporcional a la cantidad de neuronas asociativas en la capa (A). Por lo tanto, el incorporar GLCs para generar las coordenadas de la máscara y los valores umbrales en tiempo real es una opción de diseño factible para disminuir los requerimiento de memoria, sobretodo por que el implementar cada uno de los GLC requiere de unicamente de tres multiplicadores embebidos.

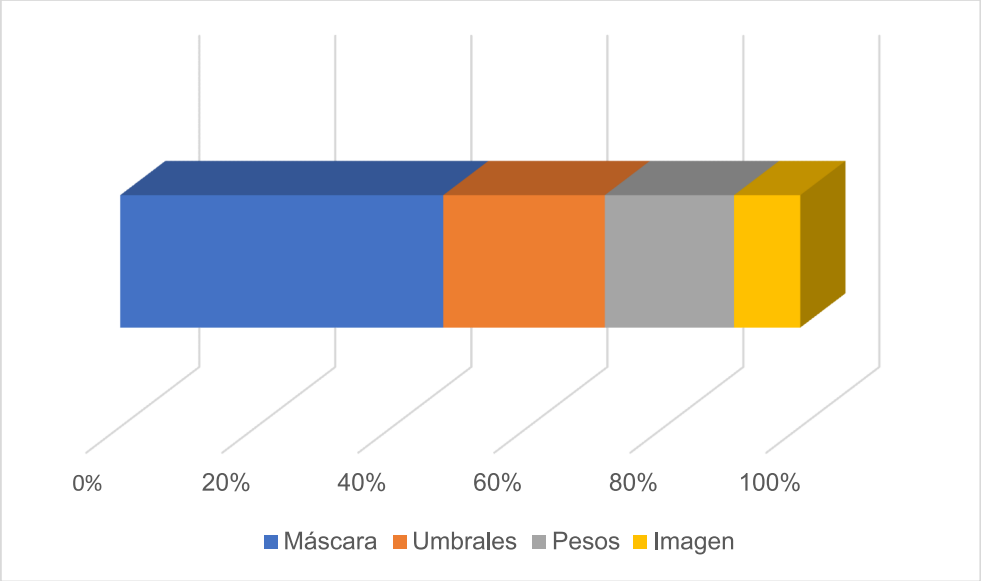


Figura 109: Porcentaje de memoria para cada tipo de dato

En la tabla 33 se muestran las especificaciones finales para el arreglo M.1 las especificaciones del uno al nueve pertenecen a la arquitectura LIRA-FPNN considerando que todos los datos se leen de memorias externas, la cantidad elementos lógicos requeridos y el valor de potencia consumida no incluyen algunos elementos extras que conforman el sistema, tales como los controladores para las memorias o los controladores u otros elementos para la adquisición de la imagen, por lo que estos valores serán mayores una vez se complete el sistema y se realice la integración final en el chip.

Por último en la figura 110 se muestra la representación esquemática de la arquitectura de LIRA-FPNN completamente conectada, contiene treinta y dos bloques de cálculo y trabaja a una frecuencia máxima de 69 [MHz], sin embargo, la frecuencia de escritura para los bloques de memoria es mayor, superando los 100 [MHz].

Tabla 33: Especificaciones finales de M.1

Núm	Núm. de Necesidad	Métrica	Imp.	Unid.	Valor final
1	13	Tamaño de la imagen	5	px	256 × 256
2	1,13	Profundidad de bits	4	bits	8
3	12	Memoria para datos	5	MB	5.4
4	10,11	Frecuencia máxima	4	MHz	69
5	2	Potencia	4	mW	148
6	4,6	Bloques de calculo	4	Cblock	32
7	4	Iteraciones	5	Ciclos	1000
8	3	Elementos lógicos	5	LE	4524
9	3	Memoria embebida BRAM	5	bits	6144
10	3	Multiplicadores por GLC	3	Num	3

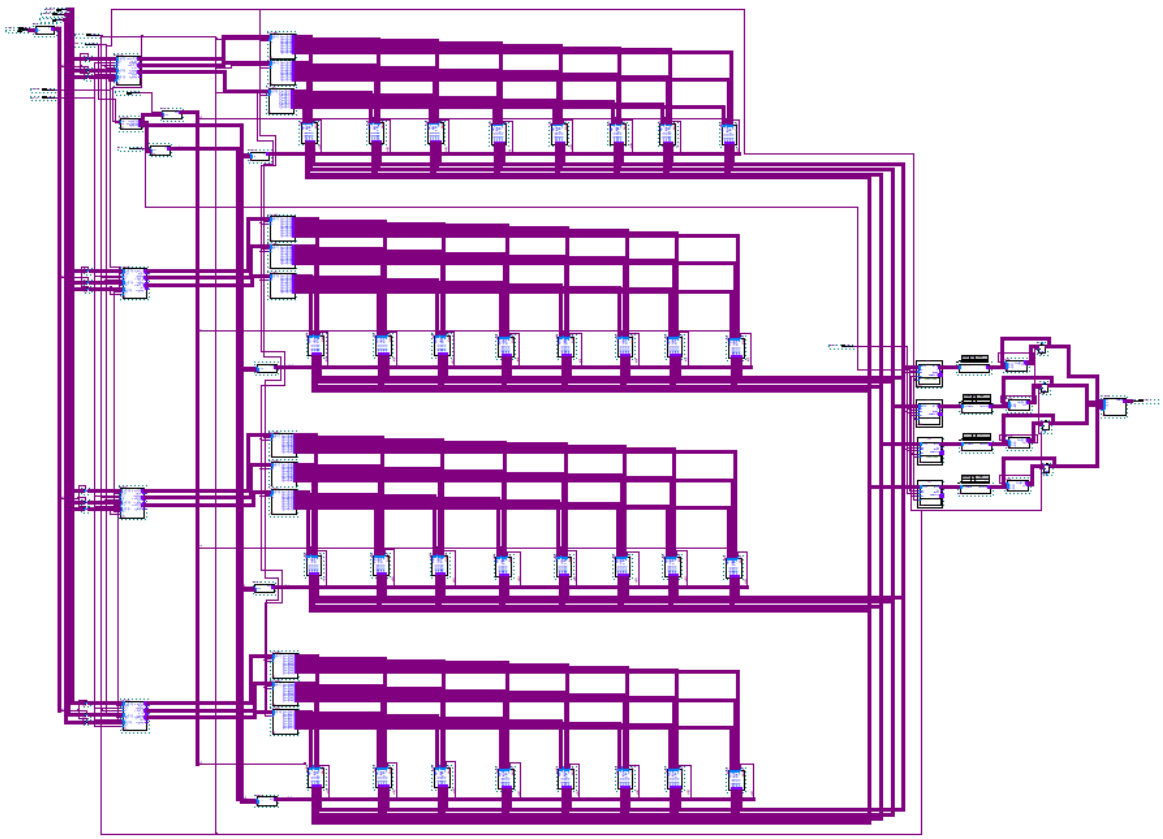


Figura 110: Estructura de la red LIRA-FPNN final

13

CONCLUSIONES

Se probo el clasificador neuronal LIRA-escala de grises en la tarea de reconocimiento de texturas del terreno marciano. El clasificador no utiliza operaciones de multiplicación o coma flotante. Esta propiedad combinada con la estructura paralela del clasificador permite su implementación en dispositivos electrónicos de descripción de hardware y alta velocidad tales como los FPGA. Se obtuvo una convergencia suficientemente rápida del proceso de entrenamiento y una alta tasa de reconocimiento del 98,9 % con la base de de imágenes experimentales ITM creada para este trabajo.

Se obtuvo el valor de los parámetros de la red más adecuados para la aplicación entre ellos las dimensiones de la ventana de escaneo que minimiza el porcentaje de error en el reconocimiento, también se demostró el comportamiento del porcentaje de reconocimiento en relación a la cantidad de imágenes utilizadas para el entrenamiento, mientras más imágenes se utilicen el sistema tendrá más información para aprender a reconocer los tipos de clases.

Se aplicó la metodología FPNA/FPNN para diseñar la implementación del clasificador neuronal LIRA en dispositivos FPGA, se realizó un diseño modular y escalable el cual esta orientado al ahorro de área manteniendo un bajo consumo de potencia. Se comprueba que el diseño LIRA-FPNN puede ser implementado prácticamente cualquier dispositivo de gama media actual debido a que sus requerimientos de memoria, elementos lógicos y multiplicadores embebidos son aceptables y relativamente bajos.

Las implementaciones de hardware de redes neuronales tienen que conciliar topologías de hardware simples con, a menudo, complejas arquitecturas neuronales, el esquema de cálculo utilizado (FPNA/FPNN) crea numerosos enlaces neuronales virtuales por medio de un conjunto limitado de enlaces de comunicación, independientemente del dispositivo, la aritmética y la estructura neuronal, lo que permite tener el poder de cálculo de los modelos neuronales estándar con un conjunto reducido de recursos neuronales fáciles de mapear directamente en hardware digital.

La tesis sirve como guía para la implementación de redes neuronales en FPGA utilizando la metodología FPNA/FPNN con entrenamiento fuera de chip. También, se presentan las bases para el uso de esta tecnología junto con los métodos de aprendizaje automático para la resolución de problemas en el área espacial.

14

TRABAJO A FUTURO

- Entrenar el clasificador neuronal LIRA con la base de imágenes ITM añadiendo diferentes tipos de distorsiones a cada imagen para analizar y mejorar el desempeño del sistema.
- Realizar la integración del segmento M.1 junto con los controladores de memoria y los módulos de adquisición de imagen en una tarjeta de desarrollo FPGA.
- Incrementar la frecuencia máxima trabajo haciendo un análisis de exhaustivo de tiempos así como aplicar metodologías de encauzamiento en los módulos que lo a meriten.
- Comparar el desempeño de la red neuronal LIRA implementada en FPGA con respecto a la implementación el lenguaje de alto nivel

ANEXOS Y APÉNDICES

A

CONCEPTOS DE VISIÓN ARTIFICIAL

CÁMARA: Dispositivo usado para capturar imágenes visuales, incluyendo, fotos y señal de video

CÁMARA INTELIGENTE: Sistema de visión completo o casi completo contenido en el cuerpo mismo de la cámara, como mínimo una cámara inteligente combina una cámara y programas relacionados con procesamiento de imágenes y visión de máquina dentro de la misma carcasa. Es funcionalmente equivalente a un procesador de visión integrado.

SENSOR DE VISIÓN: Es una cámara inteligente de gama baja. Una cámara inteligente con menos flexibilidad y capacidad de programación que generalmente está diseñada para aplicaciones menos exigentes.

PROCESADOR DE VISIÓN INTEGRADO: Configuración del equipo de visión artificial donde una cámara está conectada a una minicomputadora especializada (no una computadora personal.). A diferencia de la cámara inteligente, la potencia de la computadora para procesar imágenes es externa a la carcasa de la cámara.

SENSOR: El chip de silicón rectangular dentro de la cámara que captura y procesa los fotones y los convierte en señales eléctricas para crear una imagen. Normalmente los sensores vienen en variedad CMOS o CCD.

SENSOR CCD: Los dispositivos de carga acoplada (CCD, por sus siglas en ingles) es la mas antigua y ampliamente usada tecnología de sensores; cuentan con una sola capa de silicón cubierta con canales y tiras de aluminio, los canales actúan como barreras de pixeles y permiten a los electrones moverse línea por línea al largo de las tiras de aluminio hasta que son enviados a un capacitor que transfiere la información a dispositivos de almacenamiento; estos dispositivos son conocidos por ser generar bajo ruido, hoy son utilizados en cámaras de gama media.

SENSOR CMOS: Los sensores de semiconductor complementario de óxido metálico (CMOS, por sus siglas en inglés) son una tecnología más nueva que CCD, utiliza un chip de silicón cubierto por pixeles donde cada uno contiene su propio capacitor y amplificador. Los sensores CMOS consumen menos potencia que los sensores CCD y son capaces de leer información rápidamente; estos sensores son utilizados en las cámaras más modernas.

PIXEL O PUNTO: Es la unidad básica de recolección de luz en un sensor, cada pixel recolecta la luz y lo despliega como un color. En un monitor es la unidad más pequeña discernible que despliega información. En general, mientras más pixeles tenga el sensor de una cámara, mayor será la resolución y nitidez de la imagen que aparecerá.

MEGAPIXEL: 1 megapixel es igual a 1,000,000 pixeles cuadrados, los megapíxeles son calculados mediante la multiplicación el número de pixeles horizontales y verticales. Cada millón de pixeles es igual a un megapíxel. Por lo tanto, un sensor con 2,527,000 pixeles se consideran como un sensor de 2.5 megapíxeles.

RESOLUCIÓN: A menudo descrito en megapíxeles, la resolución es a menudo referida al número de pixeles horizontales (en miles) seguido de una letra k (e.g. 4096×2160 pixeles es una resolución 4k)

RANGO DINÁMICO: Comúnmente medido en *stops*, es la medida de la habilidad de la cámara de capturar elementos tanto en luz como en oscuridad de manera inmediata. El alto rango dinámico (HDR, por sus siglas en inglés) te muestra mayor detalle en las áreas muy oscuras o muy brillosas, a través de la recolección de información de color en esas áreas. Por ejemplo, en lugar de un cielo muy brillante que podría verse color blanco se adapta para verse azul.

PROFUNDIDAD DE BITS: Hace referencia al número de colores en el sensor o monitor de una cámara. La mayoría de las cámaras y los monitores utiliza alguna porción del espectro de color RGB (Red, green, blue), que se refiere al espacio de color general que incluye cada color perceptible. El número de bits utilizado se aplica para cada uno de los canales: rojo, verde y azul y se mide en base 2. Por lo tanto, 1bit=2^1 bits=2 colores por canal u 8 colores en total. 2 bits=$2^2 = 4$ colores para cada canal (64 en total) y así. Un color de 8-bit significa que se tienen $2^8 = 256$ variaciones de rojo, verde y azul = 16,777,216 colores en total.

FOTOGRAMA COMPLETO: En inglés es conocido como full frame, se refiere a un sensor que tiene al menos 36mm de ancho. El fotograma completo estándar para fotografía es de 36mm por 24mm, pero las camas recientes de cine con sensores de fotograma completo a menudo usan sensores mas anchos que van de 35mm a 38 mm de ancho. Un sensor de fotograma completo tiene la meta principal de trabajar perfectamente detrás de lentes de fotograma completo sin ningún factor de recorte (en acercamientos o alejamientos) y sin ningún viñeteado. (bordes obscuros en el marco)

RELACIÓN DE ASPECTO: La relación de aspecto y la resolución van de la mano, la relación de aspecto es simplemente la proporción de la resolución horizontal a la relación vertical ancho:altura (e.g 16:9 o 17:9), cuando la relación coincide con la pantalla la imagen la llenará por completo (conocido como full screen), pero cuando no, aparecerán barras negras arriba y abajo o a los lados. Dejando las barras negras, se es capaz de ver la imagen completa que fue capturada, de otro modo es necesario escalarla para que quepa, y esto puede cortar alguna porción de la imagen.

HISTOGRAMA: Es la función obtenida de una imagen, que nos indica la frecuencia de ocurrencia de los tonos de gris presentes en la imagen. Proporciona información importante como, por ejemplo, contraste y brillo de la imagen, información para la segmentación por umbrales de objetos presentes en la imagen, estadísticas de la imagen, etc.

DISPARIDAD: La disparad binocular se refiere a la diferencia en la ubicación de la imagen de un objeto visto por los ojos izquierdo y derecho, como resultado de la separación de los ojos (paralaje)

A.1 Conceptos de procesamiento de imagen

PROCESAMIENTO DIGITAL DE IMAGEN: Es el término general utilizado para el procesamiento de imágenes por computadora, algunos autores reservan este término únicamente para las operaciones en donde la entrada y la salida de información es una imagen como tal.

MEJORAMIENTO DE LA IMAGEN: Implica mejorar la calidad subjetiva de una imagen, o la detectabilidad de objetos dentro de la imagen, la información que se mejora generalmente es evidente en la imagen original pero puede no ser clara. Los ejemplos de mejora de imagen incluyen reducción de ruido, mejora de contraste, nitidez de bordes y corrección de color.

RESTAURACIÓN DE LA IMAGEN: Utiliza el conocimiento de las posibles causas de la degradación presentes en una imagen para crear un modelo, este modelo se utiliza para derivar un proceso inverso que se utiliza para restaurar la imagen; en muchos casos, la información en la imagen se ha degradado hasta el punto de ser irreconocible, por ejemplo alguna borrosidad severa.

RECONSTRUCCIÓN DE LA IMAGEN: Implica reestructurar los datos que están disponibles en una forma más útil, los ejemplos son la superresolución de la imagen (reconstrucción de una imagen de alta resolución a partir de una serie de imágenes de baja resolución) y la tomografía (reconstrucción de una sección transversal de un objeto a partir de una serie de proyecciones).

ANÁLISIS DE IMAGEN: Se refiere específicamente al uso de computadoras para extraer información de imágenes, el resultado suele ser alguna forma de medición. En el pasado, esto era casi exclusivamente para imágenes bidimensionales, aunque con el advenimiento de la microscopía confocal y otras técnicas avanzadas de imágenes, actualmente esto se ha extendido a tres dimensiones.

RECONOCIMIENTO DE PATRONES: Se refiere a la identificación de objetos basados en patrones. hay una fuerte tendencia en la utilización de enfoques estadísticos, aunque también se utilizan métodos sintácticos y estructurales.

VISIÓN POR COMPUTADORA: Utilizar un enfoque basado en modelos para el procesamiento de imágenes, los modelos matemáticos de la escena y el procesamiento de imágenes se utilizan para derivar una representación tridimensional basada en imágenes de una o dos dimensiones de una escena. El uso de modelos implícitamente proporciona una interpretación de los contenidos de las imágenes obtenidas.

Igualmente los campos del procesamiento de imágenes se distinguen según la aplicación:

VISIÓN ARTIFICIAL: Utiliza el procesamiento de imágenes como parte del sistema de control de una máquina. Las imágenes se capturan y se analizan; los resultados se utilizan directamente para controlar la máquina mientras esta realiza una tarea específica; A menudo se enfatiza el procesamiento en tiempo real.

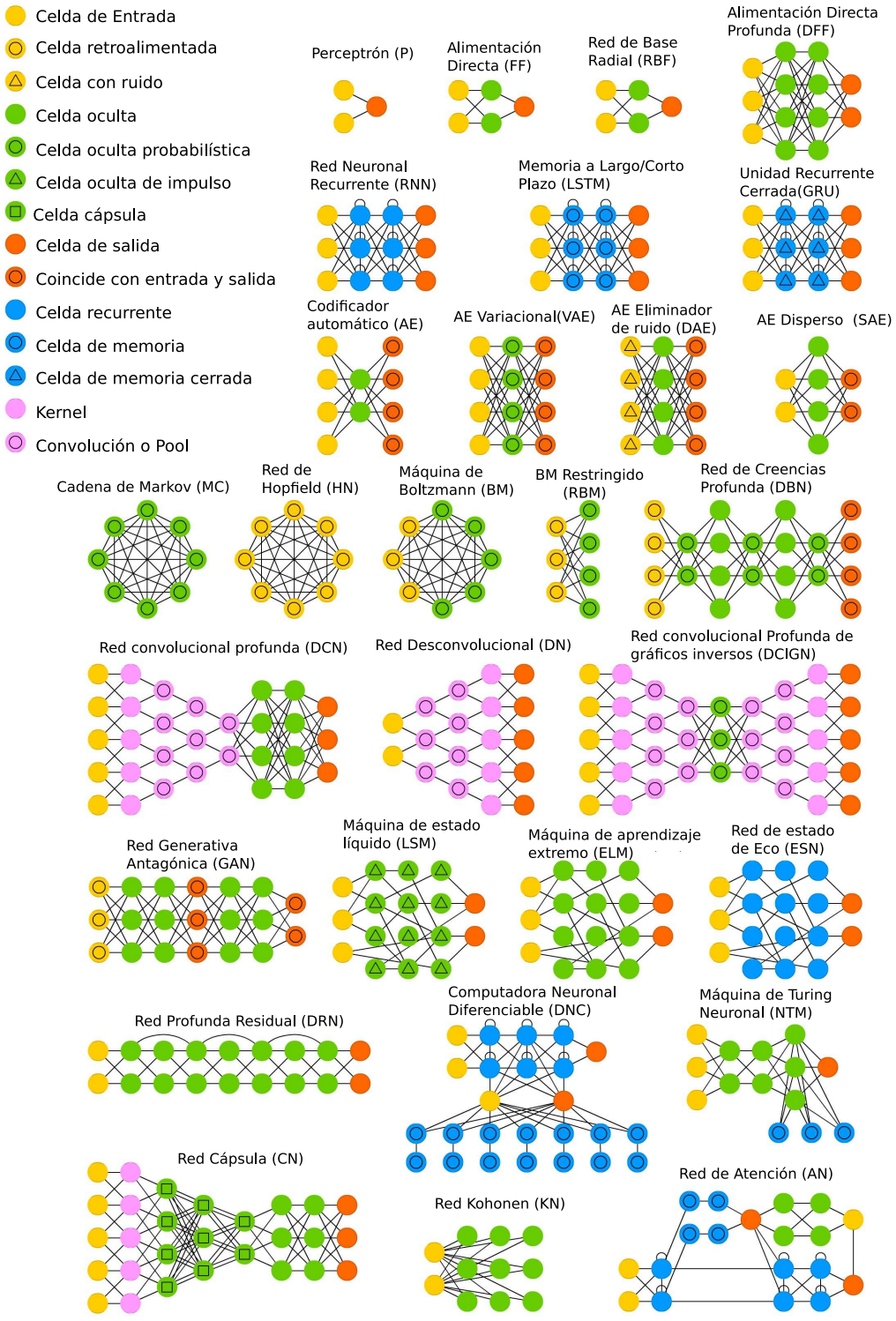
PERCEPCIÓN REMOTA: Generalmente se refiere al uso de análisis de imágenes para obtener información geográfica, ya sea mediante imágenes satelitales o fotografías aéreas.

IMAGEN MÉDICA: Abarca una amplia gama de modalidades de imagenes (rayos X, ultrasonido, resonancia magnética) relacionadas principalmente con el diagnóstico y otras aplicaciones médicas; implica tanto la reconstrucción de imágenes para crear imágenes significativas a partir de los datos sin procesar recopilados por los sensores, como el análisis de imágenes para extraer información útil.

CODIFICACIÓN DE IMAGEN Y VIDEO: Se enfoca en la compresión de una imagen o secuencia de imágenes para que ocupe menos espacio de almacenamiento o tome menos tiempo para transmitir de una ubicación a otra, la compresión es posible porque muchas imágenes contienen información significativa redundante. En el paso inverso (decodificación de imagen) la imagen completa o el video se reconstruye a partir de los datos comprimidos.

B

DIAGRAMA DE REDES NEURONALES



C

TAXONOMIA DE LOS SISTEMAS

En la vida real los sistemas existen como parte de uno o varios subsistemas; es necesario dividir estos sistemas en componentes o partes con el fin de facilitar el proceso de desarrollo. Dicho lo anterior, en este trabajo se utilizó la taxonomía de los componentes de para los sistemas espaciales propuesta en el libro *Fundamentals of Space Systems* de *Vincent L. Pisacane* [62], donde:

- **Sistema:** Es el grupo de segmentos desplegados y operados para satisfacer un objetivo de la misión, consta de hardware, software y personal operativo. Algunos ejemplos son el Sistema de Posicionamiento Global (GPS), y el Sistema Satelital de Seguimiento y Relevamiento de Datos (TDR) de la NASA.

- **Segmento:** Es un grupo de elementos que juntos constituyen un componente mayor o una función mayor del sistema. Como ejemplos se tiene una una constelación de satélites, una comunidad de usuarios o una red de rastreo y control.

- **Elemento:** Es un grupo de subsistemas que juntos realizan una función importante, varios de los cuales se integran para formar un segmento. Los ejemplos son un satélite de una constelación, un subconjunto de usuarios y la estación de seguimiento de una red.[1]

- **Subsistema:** es un grupo de componentes que realizan una función dentro de un elemento, segmento o sistema. Algunos ejemplos son el subsistema de determinación y control de orientación, el subsistema de potencia y el subsistema de control térmico. Hay que tener en cuenta que estos son llamados *sistemas* por sus desarrolladores.

- **Ensamble:** También denominado arreglo es un grupo de elementos que realizan una función para dar soporte a un subsistema. Los ejemplos relacionados con el subsistema de determinación de orientación son: el ensamble de determinación de orientación solar, el ensamble de orientación de campo magnético, el ensamble del sistema de medición inercial y el ensamble de ruedas de reacción.

- **Sub-ensamble:** Es una subdivisión funcional de un ensamble. Los ejemplos relacionados con el sistema de determinación y control de orientación incluyen los detectores de orientación solar, electrónica de orientación solar, magnetómetros y electrónica de magnetómetros, ruedas de reacción y electrónica de las ruedas de reacción.

1 Los términos **segmento** y **elemento** se emplean con poca frecuencia y por lo general no se usan cuando se hablan de una sola nave espacial o cuando hay un número limitado de estaciones de control o un número limitado de usuarios directos.

- **Parte:** Una parte es un elemento de hardware que no puede subdividirse lógicamente. Los ejemplos incluyen un chip, un diodo, una carcasa, un perno de fijación y el volante de inercia de una RW.

En la práctica, los componentes de la taxonomía anterior no se siguen estrictamente. Por ejemplo, la mayoría de los subsistemas de las naves espaciales son tan complejos que generalmente se les considera **sistemas** por si mismos; por ejemplo, el *sistema de orientación* y el *sistema de control térmico*

D

TABLAS DE PRODUCTOS FPGA

D.1 FPGA Xilinx serie-7

Artix-7 FPGAs

		Transceiver Optimization at the Lowest Cost and Highest DSP Bandwidth (1.0V, 0.95V, 0.9V)							
	Part Number	XC7A12T	XC7A15T	XC7A25T	XC7A35T	XC7A50T	XC7A75T	XC7A100T	XC7A200T
Logic Resources	Logic Cells	12,800	16,640	23,360	33,280	52,160	75,520	101,440	215,360
	Slices	2,000	2,600	3,650	5,200	8,150	11,800	15,850	33,650
	CLB Flip-Flops	16,000	20,800	29,200	41,600	65,200	94,400	126,800	269,200
Memory Resources	Maximum Distributed RAM (Kb)	171	200	313	400	600	892	1,188	2,888
	Block RAM/FIFO w/ ECC (36 Kb each)	20	25	45	50	75	105	135	365
	Total Block RAM (Kb)	720	900	1,620	1,800	2,700	3,780	4,860	13,140
Clock Resources	CMTs (1 MMCM + 1 PLL)	3	5	3	5	5	6	6	10
I/O Resources	Maximum Single-Ended I/O	150	250	150	250	250	300	300	500
	Maximum Differential I/O Pairs	72	120	72	120	120	144	144	240
Embedded Hard IP Resources	DSP Slices	40	45	80	90	120	180	240	740
	PCIe® Gen2[1]	1	1	1	1	1	1	1	1
	Analog Mixed Signal (AMS) / XADC	1	1	1	1	1	1	1	1
	Configuration AES / HMAC Blocks	1	1	1	1	1	1	1	1
	GTP Transceivers (6.6 Gb/s Max Rate)[2]	2	4	4	4	4	8	8	16
Speed Grades	Commercial Temp (C)	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2
	Extended Temp (E)	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3
	Industrial Temp (I)	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L

	Package[3], [4]	Dimensions (mm)	Ball Pitch (mm)	Available User I/O: 3.3V SelectIO™ HR I/O (GTP Transceivers)						
	CPG236	10 x 10	0.5		106 (2)		106 (2)	106 (2)			
	CPG238	10 x 10	0.5	112 (2)		112 (2)					
	CSG324	15 x 15	0.8		210 (0)		210 (0)	210 (0)	210 (0)	210 (0)	
	CSG325	15 x 15	0.8	150 (2)	150 (4)	150 (4)	150 (4)	150 (4)			
	FTG256	17 x 17	1.0		170 (0)		170 (0)	170 (0)	170 (0)	170 (0)	
	SBG484	19 x 19	0.8								285 (4)
Footprint Compatible	FGG484[5]	23 x 23	1.0		250 (4)		250 (4)	250 (4)	285 (4)	285 (4)	
	FBG484[5]	23 x 23	1.0								285 (4)
Footprint Compatible	FGG676[6]	27 x 27	1.0						300 (8)	300 (8)	
	FBG676[6]	27 x 27	1.0								400 (8)
	FFG1156	35 x 35	1.0								500 (16)

Figura 111: FPGA Artix 7

Kintex-7 FPGAs

		Optimized for Best Price-Performance (1.0V, 0.95V, 0.9V)						
	Part Number	XC7K70T	XC7K160T	XC7K325T	XC7K355T	XC7K410T	XC7K420T	XC7K480T
Logic Resources	Slices	10,250	25,350	50,950	55,650	63,550	65,150	74,650
	Logic Cells	65,600	162,240	326,080	356,160	406,720	416,960	477,760
	CLB Flip-Flops	82,000	202,800	407,600	445,200	508,400	521,200	597,200
Memory Resources	Maximum Distributed RAM (Kb)	838	2,188	4,000	5,088	5,663	5,938	6,788
	Block RAM/FIFO w/ ECC (36 Kb each)	135	325	445	715	795	835	955
	Total Block RAM (Kb)	4,860	11,700	16,020	25,740	28,620	30,060	34,380
Clock Resources	CMTs (1 MMCM + 1 PLL)	6	8	10	6	10	8	8
I/O Resources	Maximum Single-Ended I/O	300	400	500	300	500	400	400
	Maximum Differential I/O Pairs	144	192	240	144	240	192	192
Integrated IP Resources	DSP48 Slices	240	600	840	1,440	1,540	1,680	1,920
	PCIe® Gen2[1]	1	1	1	1	1	1	1
	Analog Mixed Signal (AMS) / XADC	1	1	1	1	1	1	1
	Configuration AES / HMAC Blocks	1	1	1	1	1	1	1
	GTX Transceivers (12.5 Gb/s Max Rate)	8	8	16	24	16	32	32
Speed Grades	Commercial Temp (C)	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2
	Extended Temp (E)	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3
	Industrial Temp (I)	-1, -2	-1, -2, -2L	-1, -2, -2L	-1, -2, -2L	-1, -2, -2L	-1, -2, -2L	-1, -2, -2L

	Package[2]	Dimensions (mm)	Ball Pitch (mm)	Available User I/O: 3.3V HR I/O, 1.8V HP I/Os (GTX)					
	FBG484[3]	23 x 23	1.0	185, 100 (4)	185, 100 (4)				
Footprint Compatible	FBG676[3]	27 x 27	1.0	200, 100 (8)	250, 150 (8)	250, 150 (8)		250, 150 (8)	
	FFG676	27 x 27	1.0		250, 150 (8)	250, 150 (8)		250, 150 (8)	
Footprint Compatible	FBG900[3]	31 x 31	1.0			350, 150 (16)		350, 150 (16)	
	FFG900	31 x 31	1.0			350, 150 (16)		350, 150 (16)	
	FFG901	31 x 31	1.0				300, 0 (24)		380, 0 (28)
	FFG1156	35 x 35	1.0						400, 0 (32)

Figura 112: FPGA Kintex 7

Virtex-7 FPGAs

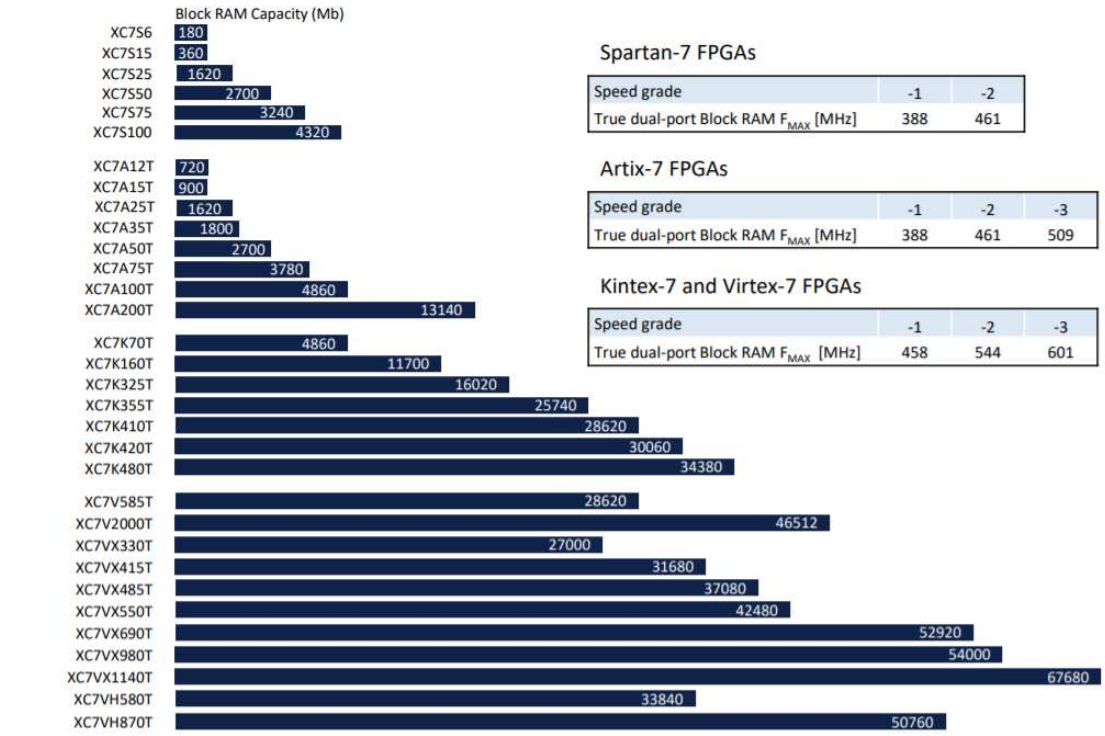
Optimized for Highest System Performance and Capacity (1.0V)

	Part Number	XC7V585T	XC7V2000T	XC7VX330T	XC7VX415T	XC7VX485T	XC7VX550T	XC7VX690T	XC7VX980T	XC7VX1140T	XC7VH580T	XC7VH870T
Logic Resources	Slices	91,050	305,400	51,000	64,400	75,900	86,600	108,300	153,000	178,000	90,700	136,900
	Logic Cells	582,720	1,954,560	326,400	412,160	485,760	554,240	693,120	979,200	1,139,200	580,480	876,160
	CLB Flip-Flops	728,400	2,443,200	408,000	515,200	607,200	692,800	866,400	1,224,000	1,424,000	725,600	1,095,200
Memory Resources	Maximum Distributed RAM (Kb)	6,938	21,550	4,388	6,525	8,175	8,725	10,888	13,838	17,700	8,850	13,275
	Block RAM/FIFO w/ ECC (36 Kb each)	795	1,292	750	880	1,030	1,180	1,470	1,500	1,880	940	1,410
	Total Block RAM (Kb)	28,620	46,512	27,000	31,680	37,080	42,480	52,920	54,000	67,680	33,840	50,760
Clocking	CMTs (1 MMCM + 1 PLL)	18	24	14	12	14	20	20	18	24	12	18
I/O Resources	Maximum Single-Ended I/O	850	1,200	700	600	700	600	1,000	900	1,100	600	300
	Maximum Differential I/O Pairs	408	576	336	288	336	288	480	432	528	288	144
Integrated IP Resources	DSP Slices	1,260	2,160	1,120	2,160	2,800	2,880	3,600	3,600	3,360	1,680	2,520
	PCIe® Gen2[1]	3	4	—	—	4	—	—	—	—	—	—
	PCIe Gen3	—	—	2	2	—	2	3	3	4	2	3
	Analog Mixed Signal (AMS) / XADC	1	1	1	1	1	1	1	1	1	1	1
	Configuration AES / HMAC Blocks	1	1	1	1	1	1	1	1	1	1	1
	GTX Transceivers (12.5 Gb/s Max Rate)[2]	36	36	—	—	56	—	—	—	—	—	—
	GTH Transceivers (13.1 Gb/s Max Rate)[3]	—	—	28	48	—	80	80	72	96	48	72
	GTZ Transceivers (28.05 Gb/s Max Rate)	—	—	—	—	—	—	—	—	—	8	16
Speed Grades	Commercial Temp (C)	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2
	Extended Temp (E)[4]	-2L, -3	-2L, -2G	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L	-2L, -2G	-2L, -2G	-2L, -2G
	Industrial Temp (I)	-1, -2	-1	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1	-1	—	—

Package[5]	Dimensions (mm)	Ball Pitch (mm)	Available User I/O: 3.3V HR I/O, 1.8V HP I/Os (GTX, GTH)									1.8V HP I/O (GTH, GTZ)	
FFG1157[6]	35 x 35	1.0	0, 600 (20, 0)		0, 600 (0, 20)	0, 600 (0, 20)	0, 600 (20, 0)		0, 600 (0, 20)				
FFG1761[6]	42.5 x 42.5	1.0	100, 750 (36, 0)		50, 650 (0, 28)		0, 700 (28, 0)		0, 850 (0, 36)				
FHG1761	45 x 45	1.0		0, 850 (36, 0)									
FLG1925	45 x 45	1.0		0, 1200 (16, 0)									
FFG1158[6]	35 x 35	1.0				0, 350 (0, 48)	0, 350 (48, 0)	0, 350 (0, 48)	0, 350 (0, 48)				
FFG1926	45 x 45	1.0							0, 720 (0, 64)	0, 720 (0, 64)			
FLG1926	45 x 45	1.0									0, 720 (0, 64)		
FFG1927[6]	45 x 45	1.0				0, 600 (0, 48)	0, 600 (56, 0)	0, 600 (0, 80)	0, 600 (0, 80)				
FFG1928	45 x 45	1.0								0, 480 (0, 72)			
FLG1928	45 x 45	1.0									0, 480 (0, 96)		
FFG1930	45 x 45	1.0					0, 700 (24, 0)		0, 1000 (0, 24)	0, 900 (0, 24)			
FLG1930	45 x 45	1.0									0, 1100 (0, 24)		
FLG1155	35 x 35	1.0										400 (24, 8)	
FLG1931	45 x 45	1.0										600 (48, 8)	
FLG1932	45 x 45	1.0											300 (72, 16)

Figura 113: FPGA Virtex 7

Block RAM Metrics



Spartan-7 FPGAs

Speed grade	-1	-2
True dual-port Block RAM F_{MAX} [MHz]	388	461

Artix-7 FPGAs

Speed grade	-1	-2	-3
True dual-port Block RAM F_{MAX} [MHz]	388	461	509

Kintex-7 and Virtex-7 FPGAs

Speed grade	-1	-2	-3
True dual-port Block RAM F_{MAX} [MHz]	458	544	601

Figura 114: Gráfica de Bram disponible en la serie 7

D.2 Altera FPGA

Intel MAX 10 FPGA Features

View device ordering codes on page 52.

PRODUCT LINE	10M02	10M04	10M08	10M16	10M25	10M40	10M50
LEs (K)	2	4	8	16	25	40	50
Block memory (Kb)	108	189	378	549	675	1,260	1,638
User flash memory[1] (KB)	12	16 – 156	32 – 172	32 – 296	32 – 400	64 – 736	64 – 736
18 x 18 multipliers	16	20	24	45	55	125	144
PLLs[2]	1, 2	1, 2	1, 2	1, 4	1, 4	1, 4	1, 4
Internal configuration	Single	Dual	Dual	Dual	Dual	Dual	Dual
Analog-to-digital converter (ADC), temperature sensing diode (TSD)[3]	-	1, 1	1, 1	1, 1	2, 1	2, 1	2, 1
External memory interface (EMIF)	Yes[4]	Yes[4]	Yes[4]	Yes[5]	Yes[5]	Yes[5]	Yes[5]

Figura 115: Gráfica de Bram disponible en la serie 7

Intel Arria 10 FPGA Features

View device ordering codes on page 51.

	PRODUCT LINE	GX 160	GX 220	GX 270	GX 320	GX 480	GX 570	GX 660	GX 900	GX 1150	GT 900	GT 1150
Resources	Part number reference	10AX016	10AX022	10AX027	10AX032	10AX048	10AX057	10AX066	10AX090	10AX115	10AT090	10AT115
	LEs (K)	160	220	270	320	480	570	660	900	1,150	900	1,150
	System logic elements (K)	210	288	354	419	629	747	865	1,180	1,506	1,180	1,506
	Adaptive logic modules (ALMs)	61,510	83,730	101,620	118,730	181,790	217,080	250,540	339,620	427,200	339,620	427,200
	Registers	246,040	334,920	406,480	474,920	727,160	868,320	1,002,160	1,358,480	1,708,800	1,358,480	1,708,800
	M20K memory blocks	440	588	750	891	1,438	1,800	2,133	2,423	2,713	2,423	2,713
	M20K memory (Mb)	9	11	15	17	28	35	42	47	53	47	53
	MLAB memory (Mb)	1.0	1.8	2.4	2.8	4.3	5.0	5.7	9.2	12.7	9.2	12.7
	Hardened single-precision floating-point multipliers/adders	156/156	191/191	830/830	985/985	1,368/1,368	1,523/1,523	1,688/1,688	1,518/1,518	1,518/1,518	1,518/1,518	1,518/1,518
	18 x 19 multipliers	312	382	1,660	1,970	2,736	3,046	3,376	3,036	3,036	3,036	3,036
	Peak fixed-point performance (GMACS)[1]	343	420	1,825	2,167	3,010	3,351	3,714	3,340	3,340	3,340	3,340
	Peak floating-point performance (GFLOPS)	140	172	747	887	1,231	1,371	1,519	1,366	1,366	1,366	1,366
Clocks, Maximum I/O Pins, and Architectural Features	Global clock networks	32	32	32	32	32	32	32	32	32	32	32
	Regional clocks	8	8	8	8	8	8	16	16	16	16	16
	I/O voltage levels supported (V)	1.2, 1.25, 1.35, 1.8, 2.5, 3.0										
	I/O standards supported	3 V I/O pins only: 3 V LVTTL, 2.5 V CMOS DDR and LVDS I/O pins: POD12, POD10, Differential POD12, Differential POD10, LVDS, RSDS, mini-LVDS, LVPECL All I/Os: 1.8 V CMOS, 1.5 V CMOS, 1.2 V CMOS, SSTL-135, SSTL-125, SSTL-18 (I and II), SSTL-15 (I and II), SSTL-12, HSTL-18 (I and II), HSTL-15 (I and II), HSTL-12 (I and II), HSUL-12, Differential SSTL-135, Differential SSTL-125, Differential SSTL-18 (I and II), Differential SSTL-15 (I and II), Differential SSTL-12, Differential HSTL-18 (I and II), Differential HSTL-15 (I and II), Differential HSTL-12 (I and II), Differential HSUL-12										
	Maximum LVDS channels (1.6 G)	120	120	168	168	222	324	270	384	384	312	312
	Maximum user I/O pins	288	288	384	384	492	696	696	768	768	624	624
	Transceiver count (17.4 Gbps)	12	12	24	24	36	48	48	96	96	72	72
	Transceiver count (25.78 Gbps)	–	–	–	–	–	–	–	–	–	6	6
	PCI Express hardened IP blocks (Gen3 x8)	1	1	2	2	2	2	2	4	4	4	4
	Maximum 3 V I/O pins	48	48	48	48	48	48	48	–	–	–	–
	Memory devices supported	DDR4, DDR3, DDR2, QDR IV, QDR II+, QDR II+ Xtreme, LPDDR3, LPDDR2, RLDRAM 3, RLDRAM II, LLDRAM II, HMC										

Figura 116: Gráfica de Bram disponible en la serie 7

Intel® Stratix® 10 GX/SX Product Table

intel

	PRODUCT LINE	GX 400 SX 400	GX 650 SX 650	GX 850 SX 850	GX 1100 SX 1100	GX 1650 SX 1650	GX 2100 SX 2100	GX 2500 SX 2500	GX 2800 SX 2800	GX 1660	GX 2110	GX 10M
Resources	Logic elements (LEs)[1]	378,000	612,000	841,000	1,325,000	1,624,000	2,005,000	2,422,000	2,753,000	1,679,000	2,073,000	10,200,000
	Adaptive logic modules (ALMs)	128,160	207,360	284,960	449,280	550,540	679,680	821,150	933,120	569,200	702,720	3,466,080
	ALM registers	512,640	829,440	1,139,840	1,797,120	2,202,160	2,718,720	3,284,600	3,732,480	2,276,800	2,810,880	13,864,320
	Hyper-Registers from Intel® Hyperflex™ FPGA architecture	Millions of Hyper-Registers distributed throughout the monolithic FPGA fabric										
	Programmable clock trees synthesizable	Hundreds of synthesizable clock trees										
	M20K memory blocks	1,537	2,489	3,477	5,461	5,851	6,501	9,963	11,721	6,162	6,847	12,950
	M20K memory size (Mb)	30	49	68	107	114	127	195	229	120	134	253
	MLAB memory size (Mb)	2	3	4	7	8	11	13	15	9	11	55
	Variable-precision digital signal processing (DSP) blocks	648	1,152	2,016	2,592	3,145	3,744	5,011	5,760	3,326	3,960	3,456
	18 x 19 multipliers	1,296	2,304	4,032	5,184	6,290	7,488	10,022	11,520	6,652	7,920	6,912
	Peak fixed-point performance (TMACS)[2]	2.6	4.6	8.1	10.4	12.6	15.0	20.0	23.0	13.3	15.8	13.8
	Peak floating-point performance (TFLOPS)[3]	1.0	1.8	3.2	4.1	5.0	6.0	8.0	9.2	5.3	6.3	5.5
I/O and Architectural Features	Secure device manager	AES-256/SHA-256 bitstream encryption/authentication, physically unclonable function (PUF), ECDSA 256/384 boot code authentication, side channel attack protection										–
	Hard processor system[4]	Quad-core 64-bit ARM* Cortex*-A53 up to 1.5 GHz with 32KB I/D cache, NEON coprocessor, 1 MB L2 Cache, direct memory access (DMA), system memory management unit, cache coherency unit, hard memory controllers, USB 2.0 x2, 1G EMAC x3, UART x2, SPI x4, I2C x5, general purpose timers x7, watchdog timer x4								–	–	–
		SX 400	SX 650	SX 850	SX 1100	SX 1650	SX 2100	SX 2500	SX 2800			
	Maximum user I/O pins	374	392	688	688	704	704	1160	1160	688	688	2,304
	Maximum LVDS pairs 1.6 Gbps (RX or TX)	192	192	336	336	336	336	576	576	336	336	1152[5]
	Total full duplex transceiver count	24	24	48	48	96	96	96	96	48	48	48
	GXT full duplex transceiver count (up to 28.3 Gbps)	16	16	32	32	64	64	64	64	32	32	–
	GX full duplex transceiver count (up to 17.4 Gbps)	8	8	16	16	32	32	32	32	16	16	48
	PCI Express* (PCIe*) hard intellectual property (IP) blocks (Gen3 x16)	1	1	2	2	4	4	4	4	2	2	4[6]
	Memory devices supported	DDR4, DDR3, DDR2, DDR, QDR II, QDR II+, RLDRAM II, RLDRAM 3, HMC, MoSys										

Figura 117: Gráfica de Bram disponible en la serie 7

E

GENERADOR LINEAL CONGRUENCIAL

Cualquiera que considere métodos
aritméticos para producir dígitos aleatorios
está, por supuesto, en pecado mortal

— John von Neumann:1951

Uno de los generadores de números aleatorios más populares que se utiliza en la actualidad es el generador lineal congruencial (GLC), introducido por primera vz en 1949 por D.H. Lehmer. El GLC es uno de los generadores más más estudiados y necesita de cuatro parámetros para trabajar:

M,	el módulo;	$0 < M$
a,	el multiplicador;	$2 \leqslant a < M$
c,	el incremento;	$0 \leqslant c < M$
X_0,	el valor inicial;	$0 \leqslant X_0 < M$

La secuencia deseada de números aleatorios $\langle X_0 \rangle$ se obtiene mediante la siguiente ecuación:

$$X_{n+1} = (aX_n + c) \quad \text{mód } M, \qquad n \geqslant 0 \tag{E.18}$$

$$u_n = \frac{X_n}{M}, 0 \leqslant X_n < M \tag{E.19}$$

A la ecuación E.18 se le llama secuencia congruencial lineal [**knuth2**]. La operación $\mathrm{mod} m$ es lo que le da al método su carácter aleatorio. Sin embargo la secuencia de números obtenidos no siempre es aleatoria para todos los valores de m, a, c y X_0, esto debido a que las secuencias congruenciales siempre entran en un ciclo repetitivo, cuando el total de números generados pertenece a un conjunto finito este ciclo es denominado periodo.

E.0.1 *Parámetros del GLC*

Una secuencia útil de números aleatorios debe tener un periodo relativamente largo, para lograr esto es necesario seleccionar los valores de sus parámetros cuidadosamente.

SELECCIÓN DEL INCREMENTO: El caso especial $C = 0$ permite que el procesos de generación de números sea un poco más rápido que cuando $C \neq 0$ ya que se ahorra la operación de adición. No obstante, la restricción $C = 0$ reduce la duración

del periodo de la secuencia, pero aún así sería posible hacer que el periodo sea relativamente largo. El método original de Lehmer tenia $c = 0$. No obstante mencionó en sus trabajos $C \neq 0$ como una posibilidad.

Los términos *metodo congruencial multiplicativo* y *metodo congruencial mixto* son usados por varios autores para denotar secuencias congruentes lineales con $c = 0$ y $c \neq 0$ respectivamente.

SELECCIÓN DEL MÓDULO: Es deseable que m sea bastante grande, ya que el periodo no puede tener mas de m elementos. Otro factor que influye en la elección de m es la velocidad de generación: se desea elegir un valor para el calculo rápido de $(aX_n + c) \mod m$, pero la división es una operación relativamente lenta y se puede evitar si m tiene un valor especialmente conveniente, tal como lo es el tamaño de palabra de la computadora utilizada.

Sea w el tamaño de palabra de la computadora, es decir 2^e en una computadora binaria, el resultado de la operación de adición generalmente se da modulo w, excepto en maquinas complementarias, y la multiplicación mod w también es bastante simple ya que el resultado es la mitad inferior del producto, por lo tanto el siguiente programa calcula la cantidad $(aX_n + c) \mod M$ eficientemente.

```
LDA  A   rA <- a
MUL  X   rAX <- (rA)X
SLAX 5   rA <- rAXmod w
ADD  C   rA <- (rA+c)mod w
```

A veces es conveniente utilizar $M = w \pm 1$ en lugar de $M = w$, la razón es que cuando $M = w$ los dígitos del lado derecho del punto decimal son mucho menos aleatorios que los dígitos del lado izquierdo obtenidos con la ecuación E.20

$$Y_n = X_n \mod d \tag{E.20}$$

*donde d es un divisor de M

Por ejemplo si $M = w = 2^e$ los 4 bits menos significativos de X_n son los números:

$$Y_n = X_n \mod 2^4$$

En este caso los 4 bits menos significativos forman una secuencia congruencial que tiene un periodo de longitud de 16 o menos (2^4), del mismo modo que los 5 bits menos significativos tienen un periodo de 32 (2^5) como máximo; y si solo se toma el bit menos significativo este tendría valor constante. como ejemplo si $w = 2^{35}$ y $M = 2^{35} - 1$ los bits de orden inferior se comportarían tan aleatoriamente como los de orden superior. Otra alternativa para esto es hacer que m sea el mayor numero primo menor que w.

SELECCIÓN DEL MULTIPLICADOR: Se desea escoger un multiplicador que produzca un periodo de longitud máxima. Un periodo largo es esencial para cualquier secuencia que se utilizará como fuente de números aleatorios. De hecho, se espera

que el periodo contenga más números de los que se utilizaran en la aplicación.

Aunque m representa un límite para la longitud del periodo los valores de los otros parámetros pueden provocar que no sea posible alcanzar la longitud máxima del periodo; por lo que surge la pregunta de si existe un valor de (a) con la cual sea posible alcanzar la longitud máxima de m.

Para responder a esta pregunta se toma el caso cuando $a = c = 1$, con esto la secuencia es simplemente $X_{n+1} = (X_n + 1) \mod m$ lo cual tiene periodo de longitud m, pero no se comporta de forma aleatoria.

Caracterizando los valores de los parámetros resulta que cuando m es producto de primos distintos, solo $a = 1$ producirá el periodo completo, pero cuando m es divisible por una alta potencia de algún número primo, hay una latitud considerable en la elección del valor de a.

El siguiente teorema explica cuando se alcanza el periodo máximo:

Teorema 1. *Un GLC definid por los parámetros (a,c,M) tiene periodo máximo de longitud M si y solo si:*

i) $MCD(c, M) = 1$;

ii) $a = 1, (\mod p)$ *por cada primo* p *que divide a* M

iii) $a = 1, (\mod 4)$ *si* M *es múltiplo de 4*

Considerando el caso especial de los generadores multiplicativos puros cuando $c = 0$ aunque el proceso de generación de números aleatorios es ligeramente más rápido **no** es posible lograr la longitud máxima del periodo.

Teorema 2. *Un GLC con los parámetros (a,o,M) tiene periodo de longitud máxima* $\lambda(M)$ *si y solo si*

i) $MCD(X_0, M) = 1$;

ii) a *es un elemento primitivo del módulo de* M

Los GLC más eficientes tienen un módulo igual a una potencia de 2, siendo más frecuentes $M = 2^{32}$ o $M = 2^{64}$, porque esto permite que el operador módulo opere tan solo truncando todos los bits excepto los bits más cercanos a la derecha (de hecho, esto puede lograrse simplemente al no computar en absoluto los bits más significativos).

La siguiente tabla muestra los parámetros de varios GLC comúnmente usados mediante la función rand() presente en las bibliotecas de varios compiladores.
Código en lenguaje de alto nivel C, Devuelve 10 números aleatorios del 0 al 100 utilizando la funcion srand().

Tabla 34: Parámetros de uso común

Fuente	M	(multiplicador) a	(incremento) c
Numerical Recipes	2^{32}	166452	1013904223
Borland C/C++	2^{32}	22695477	1
Borland Delphi, Virtual Pascal	2^{32}	134775813	1
Microsoft Visual/Quick C/C++	2^{32}	$214013(343FD_{16})$	$2531011(269EC3_{16})$
C++11s	$2^{32}-1$	48271	0
Microsoft Visual Basic (6 o anterior)	2^{24}	$1140671485(43FD43FD_{16})$	$12820163(C39EC3_{16})$
ANSI C: Watcom, Digital Mars, CodeWarrior, IBM VisualAge C/C++ C99,C11: sugerencia en ISO/IEC 9899	2^{32}	1103515245	12345
MMIX by Donald Knuth	2^64	6364136223846793005	1442695040888963407

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
main(){
        int c,n;
        srand(time(NULL));
        for(c=0; c<=10; c++){
                n = rand() %100 + 1;
                printf("%d\n", n); }
                return 0; }
```

Cuando el módulo m es constante, incluso cuando hay una instrucción de división de hardware, puede ser más rápido tomar el módulo directamente que usar la instrucción de división. Estos trucos se vuelven aún más valiosos en máquinas sin una instrucción de división de hardware o donde los números involucrados están fuera de rango. Hay trucos comunes para verificar la divisibilidad, pero a final de cuantas todos son en realidad casos especiales de la siguiente regla general:

- a es representado en un número en base b
- $a \mod M = ((b \mod M)(a/b) + (a \mod b)) \mod M$

Calcular el módulo para potencias de dos es trivial en una computadora binaria, el término ($b \mod M$) es cero, por lo que simplemente tomamos el módulo examinando los bits menos significativos de la representación binaria:

$$a \mod 2^i = a\&(2^i - 1)$$

Por lo tanto para $a \mod 2$ se usa a & 1, para $a \mod 4$, se usa a & 3 y para a mód 8 se usa a & 7; donde & representa la operación lógica AND

F

EL MITO DE ORFEO

La lira es un instrumento muy representativo de la cultura griega, basta con revisar la literatura para notar que este instrumento acompaña a varios personajes mitológicos[1] y es el símbolo de la cultura y música de esta civilización. La mitología griega le confiere al dios Hermes la creación de este instrumento musical; las liras antiguas eran denominadas Chelys debido a que sus cajas de resonancia eran hechas con caparazones de tortuga (cheloné). Sin embargo, las primeras liras solo poseían tres cuerdas, grave, media y aguda, las cuales recibieron el nombre de las tres musas de la ciudad de Delfos: Nete, Mese e Hípate.

Orfeo (en griego Ορφεύσ), es un personaje de la mitología griega, hijo de la musa Calíope, fue el mejor y más famoso músico y poeta de su tiempo. El instrumento que figuró como su atributo principal fue la **lira**, la cual recibió de Hermes a cambio del caduceo.[2] Cuando Orfeo tocaba sus melodías era capaz de conmover a cualquier ser vivo, incluso los arboles inclinaban las ramas a su paso, él hacia que cualquier fiera se amansara y que los hombres olvidaran su agresividad.

Orfeo realizó muchas profesas y participó en la expedición de Jasón y los argonautas, donde neutralizó con su música el canto de las sirenas, quienes atraían a los marineros haciendo que los barcos se estrellasen contra las rocas. Pero su más grande hazaña, es sin duda su descenso a los infiernos en busca de su esposa Eurídice, quien murió a causa de la mordedura de una serpiente durante su noche de bodas; Orfeo desesperado, se dirigió al inframundo donde sorteo todos los obstáculos, logró dormir con su música al gran Can Cerbero y conmovió con su música a el barquero Caronte e incluso a Hades.

Gracias a esta gran proeza, se le permitió a Eurídice regresar al mundo de los vivos si cumplía una condición: debía seguir a su esposo en la obscuridad de camino al reino de los vivos, pero su esposo jamas deberá volver su mirada hacia ella durante todo el camino. Desgraciadamente, antes de llegar a la luz y a su libertad, a Orfeo le asaltó el temor de perder nuevamente a su amada e incapaz de resistirse volvió la cara hacia ella y en el momento en que sus ojos se posaron sobre su mujer, esta desapareció para siempre.

En este trabajo se decidió asignarle al sistema diseñado un nombre clave con fines de identificación, el nombre elegido fue Orfeo, ya que este se relaciona bastante bien con el nombre LIRA, que es el nombre del clasificador neuronal en el cual esta basado el sistema de reconocimiento. Así mismo, los subsistemas que lo conforman fueron designados como N.1 y M.1, haciendo alusión a las musas de las dos primera cuerdas de la lira antigua: Nete y Mese.

1 Apolo, Terpsícore, Orfeo, Lino, Anfión, Demódoco, etc.
2 Vara con dos serpientes enrolladas, símbolo del comercio

Figura 118: Orfeo conduciendo a Eurídice fuera del infierno (1861) de Camille Corot

CRÉDITOS FOTOGRÁFICOS

- Imagen 1: Indianapolis Museum of Art at Newfields
- Imagen 2: NASA/JPL/nasasearch/images
- Imagen 3: NASA/JPL/nasasearch/images
- Imagen 4: NASA/JPL/nasasearch/images
- Imagen 5: NASA/photojournal
- Imagen 6: NASA/JPL/Malin Space Science Systems
- Imagen 7: NASA/JPL/University of Arizona
- Imagen 8: NASA/JPL
- Imagen 9: NASA/USGS University of Arizona
- Imagen 11 a) y b): NASA/JPL-Caltech/MSSS
- Imagen 11 c): NASA/JPL-Caltech/University of Arizona/Cornell
- Imagen 11 d): NASA / JPL-Caltech / MSSS
- Imagen 11 e): NASA/JPL-Caltech/Cornell/USGS
- Imagen 11 f): NASA / JPL-Caltech / Cornell / USGS
- Imagen 11 g): NASA/JPL-Caltech/Cornell
- Imagen 11 h): NASA/JPL-Caltech/MSSS
- Imagen 13 a): NASA/GSFC/Arizona State University
- Imagen 13 b): The von Hoerner & Sulger GmbH (vH&S)
- Imagen 13 c): JAXA/SELENE-II
- Imagen 13 d): NASA/DARPA
- Imagen 13 e): NASA/MER
- Imagen 14: NASA/JPL-Caltech
- Imagen 20 NASA/JPL-Caltech
- Imagen 21: NASA/JPL-Caltech
- Imagen 24: Servicios postales de Kiev
- Imagen 25: Cyberneticzoo/TAIR [29]
- Imagen 26: Neural Networks and Micromechanics [44]

BIBLIOGRAFÍA

[1] N.M Amosov. *Modeling of Thinking and the Mind*. Macmillan Education, 1967. ISBN: 978-1-349-00640-3.

[2] D Anguita, S Bencetti, A De Gloria, G Parodi, D Ricci y S Ridella. «FPGA implementation of high precision feedforward networks». En: *Proc. MicroNeuro* (1997), págs. 240-243.

[3] E Asphaug y Jekan Thangavelautham. «Asteroid Regolith Mechanics and Primary Accretion Experiments in a Cubesat». En: *LPSC Abstract* (2014).

[4] S L Bade y B L Hutchings. «FPGA based stochastic neural networks implementation». En: *Proceedings of IEEE Workshop on FPGA's for Custom Computing Machines*. 1994, págs. 189-198. DOI: 10.1109/FPGA.1994.315612.

[5] Donald Bailey. *Design for Embedded Image Processing on FPGAs*. IEEE, 2011. ISBN: 978-0-470-82850-2.

[6] Nadine Barlow. *Mars: An introduction to its interior, surface and atmosphere*. Cambridge University Press, 2008.

[7] Tetyana Baydyk y Ernst Kussul. «New Application of LIRA Neural Network». En: October (2015).

[8] Mark Betancourt. *Mars, Underground*. 2016. URL: https://www.airspacemag.com/space/mars-caves-180959123/ (visitado 30-07-2019).

[9] M Bohrn, L Fujcik y R Vrba. «Field Programmable Neural Array for feedforward neural networks». En: *2013 36th International Conference on Telecommunications and Signal Processing (TSP)*. 2013, págs. 727-731. DOI: 10.1109/TSP.2013.6614033.

[10] Mario Chacón Murguía, Rafael Sandoval Rodríguez y Javier Vega Pineda. *Percepción visual aplicada a la robotica*. Alfaomega, 2015.

[11] XiaoQi Chen, Y.Q. Chen y J.G. Chase, eds. *Mobile Robots - State of the Art in Land, Sea, Air, and Collaborative Missions*. In-Tech, 2009.

[12] Charles E. Cox y W. Ekkehard Blanz. «GANGLIONA Fast Field-Programmable Gate Array Implementation of a Connectionist Classifier». En: *IEEE Journal of Solid-State Circuits* 27.3 (1992), págs. 288-299. ISSN: 1558173X. DOI: 10.1109/4.121550.

[13] M van Daalen, P Jeavons y J Shawe-Taylor. «A stochastic neural architecture that exploits dynamically reconfigurable FPGAs». En: *[1993] Proceedings IEEE Workshop on FPGAs for Custom Computing Machines*. 1993, págs. 202-211. DOI: 10.1109/FPGA.1993.279462.

[14] *Diccionario Merriam-Webster*. URL: www.merriam-webster.com (visitado 20-09-2006).

[15] Richard O. Duda, E. Hart Peter y David G. Stork. *Pattern classification*. 2001.

[16] J G Eldredge y B L Hutchings. «RRANN: a hardware implementation of the backpropagation algorithm using reconfigurable FPGAs». En: *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*. Vol. 4. 1994, 2097-2102 vol.4. DOI: 10.1109/ICNN.1994.374538.

[17] Alex Ellery. *PLANETARY ROVERS*. Ottawa: Springer, 2016. ISBN: 978-3-642-03259-2. DOI: 10.1007/978-3-642-03259-2.

[18] Tara Estlin, Benjamin Bornstein, Daniel Gaines, Robert Anderson, David Thompson, Michael Burl, Richard Castaño y Michele Judd. «AEGIS automated targeting for MER opportunity rover». En: *ACM Transactions on Intelligent Systems and Technology (TIST)* 3 (2012). DOI: 10.1145/2168752.2168764.

[19] Ethan Farquhar, Christal Gordon y Paul Hasler. «A field programmable neural array». En: *Proceedings - IEEE International Symposium on Circuits and Systems*. 2006, 4 pp. -4117. DOI: 10.1109/ISCAS.2006.1693534.

[20] Hannes F. Paulus Georg Glaeser. *The Evolution of the Eye*. Springer International, 2015. ISBN: 978-3-319-17476-1.

[21] B Girau. «Building a 2D-compatible multilayer neural network». En: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*. Vol. 2. 2000, 59-64 vol.2. DOI: 10.1109/IJCNN.2000.857875.

[22] B Girau. «Digital hardware implementation of 2D compatible neural networks». En: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*. Vol. 3. 2000, 506-511 vol.3. DOI: 10.1109/IJCNN.2000.861359.

[23] Bernard Girau y Arnaud Tisserand. «MLP computing and learning on FPGA using on-line arithmetic». En: (1999).

[24] Michael Gschwind, Valentina Salapura y O Maischberger. «A Generic Building Block For Hopfield Neural Networks With On-chip Learning». En: 1996, págs. 49-52. ISBN: 0-7803-3073-0. DOI: 10.1109/ISCAS.1996.598474.

[25] J Guo, Z Yan, L Hou, A Lv y N Jiang. «Research on Field Programmable Neuron Array (FPNA) Based on Reusable ANN Neurons». En: *2018 14th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*. 2018, págs. 1-3. DOI: 10.1109/ICSICT.2018.8565768.

[26] Gregory Hallock Smith, Edward Hagerott, Lawrence Scherr, Kenneth Herkenhoff y James Bell III. «Optical designs for the Mars '03 rover cameras». En: *The International Society for Optical Engineering* (2001). DOI: 10.1117/12.449558.

[27] Donald O. Hebb. *The Organization of Behavior: A Neuropsychological Theory*. Nueva york: Wiley, 1949.

[28] H Hikawa. «Frequency-based multilayer neural network with on-chip learning and enhanced neuron characteristics». En: *IEEE Transactions on Neural Networks* 10.3 (1999), págs. 545-553. ISSN: 1941-0093. DOI: 10.1109/72.761711.

[29] Reuben Hoggett. *cyberneticzoo*. 2010. URL: https://bit.ly/2PCNp64 (visitado 21-06-2019).

[30] P W Hollis, J S Harper y J J Paulos. «The Effects of Precision Constraints in a Backpropagation Learning Network». En: *Neural Computation* 2.3 (1990), págs. 363-373. ISSN: 0899-7667. DOI: 10.1162/neco.1990.2.3.363.

[31] JPL. *PDS Image Atlas*. 2019. URL: https://pds-imaging.jpl.nasa.gov/search/.

[32] Khan-Academy. *El potencial de membrana*. URL: https://bit.ly/2UWlMbF (visitado 20-11-2019).

[33] Joost Nico Kok. *Artificial Intelligence*. Encyclopedia of life support systems, 2009.

[34] K Kollmann, K . Riemschneider y H C Zeidler. «On-chip backpropagation training using parallel stochastic bit streams». En: *Proceedings of Fifth International Conference on Microelectronics for Neural Networks*. 1996, págs. 149-156. DOI: 10.1109/MNNFS.1996.493785.

[35] M Krcma, J Kastil y Z Kotásek. «Mapping Trained Neural Networks to FPNNs». En: *2015 IEEE 18th International Symposium on Design and Diagnostics of Electronic Circuits Systems*. 2015, págs. 157-160. DOI: 10.1109/DDECS.2015.50.

[36] David Kriesel. *A Brief Introduction to Neural Networks*. 2007.

[37] Ernst Kussul y Tetyana Baidyk. «Neural Random Threshold Classifier in OCR Application». En: *Proceedings of the Second All-Ukrainian Intern* (1994).

[38] Ernst Kussul y Tetyana Baidyk. «Flat image recognition in the process of microdevice assembly». En: *Pattern Recognition Letters* 25 (2004).

[39] Ernst Kussul y Tetyana Baidyk. «Improved method of handwritten digit recognition tested on MNIST database». En: *Image and Vision Computing* 22 (2004).

[40] Ernst Kussul y Tetyana Baidyk. «Texture Recognition with Random Subspace Neural Classifier». En: *WSEAS Transactions on Circuits and Systems* 4.4 (2005).

[41] Ernst Kussul y Tetyana Baidyk. «Neural classifier for larvae recognition». En: *The 2010 International Joint Conference on Neural Networks (IJCNN)* (2010). DOI: 10.1109/IJCNN.2010.5596349.

[42] Ernst Kussul, Tetyana Baidyk y T.N. Lukovitch. «Rosenblatt Perceptrons for Handwritten Digit Recognition». En: *International Joint Conference on Neural Networks* (2001).

[43] Ernst Kussul, Tetyana Baidyk, T.N. Lukovitch y V.V. Rachkovskij. «Adaptive high performance classifier based on random threshold neurons». En: *Cybernetics and Systems* 94 (1994).

[44] Ernst Kussul, Tetyana Baidyk y Donald Wunch. *Neural Networks and Micromechanics*. Springer, 2010. DOI: 10.1007/978-3-642-02535-8.

[45] Emily Lakdawalla. *The Design and Engineering of Curiosity*. Springer, 2018.

[46] Jean-Claude Latombe. *Robot Motion Models*. 1991, págs. 140-173. DOI: 10.4018/978-1-4666-2104-6.ch005.

[47] Steven Leibson. «Baby You Can Drive My Rover». En: (). URL: https://www.eejournal.com/article/baby-you-can-drive-my-rover/.

[48] Jack Lightholder, Andrew Thoesen, Eric Adamson, Jeremy Jakubowski, Ravi Nallapu, Sarah Smallwood, Laksh Raura, Andrew Klesh, Erik Asphaug y Jekan Thangavelautham. «Asteroid Origins Satellite (AOSAT) I: An On-orbit Centrifuge Science Laboratory». En: *Acta Astronautica* 133 (2017), págs. 81-94. ISSN: 0094-5765. DOI: https://doi.org/10.1016/j.actaastro.2016.12.040. URL: http://www.sciencedirect.com/science/article/pii/S0094576515301521.

[49] P Lysaght, J Stockwood, J Law y Dereje Girma. «Artificial Neural Network Implementation on a Fine-Grained FPGA». En: (2002).

[50] MARCO AURELIO NUÑO MAGANDA. «A high performance hardware architecture for multilayer spiking neural networks». Tesis doct. Instituto Nacional de Astrofísica, Óptica y Electrónica, 2009.

[51] Oleksandr Makeyev y Tetyana Baidyk. «Limited receptive area neural classifier for texture recognition of mechanically treated metal surfaces». En: *Neurocomputing* 71 (2008).

[52] J. N. Maki, J. F. Bell III y K. E. Herkenhoff. «Mars Exploration Rover Engineering Cameras». En: *Journa of geophysical Research* 108 (2003). DOI: 10.1029/2003JE002077.

[53] J. Maki, D. Thiessen y A.Pourangi. «The Mars Science Laboratory Engineering Cameras». En: *Springer Science+Business Media* (201). DOI: 10.1007/s11214-012-9882-4.

[54] Amy Mcgovern y Kiri Wagstaff. «Machine learning in space: Extending our reach». En: *Machine Learning* 84 (2011), págs. 335-340. DOI: 10.1007/s10994-011-5249-4.

[55] Marzieh Moradi, Mohammad Ali Poormina y Farbod Razzazi. «FPGA Implementation of Feature Extraction and MLP Neural Network Classifier for Farsi Handwritten Digit Recognition». En: *Third UKSim European Symposium on Computer Modeling and Simulation* (2009).

[56] NASA. *MEP Raw Images*. URL: https://mars.nasa.gov/msl/multimedia/raw-images/ (visitado 30-11-2019).

[57] NASA. *Mars Curiosityn Rover*. URL: https://mars.nasa.gov/msl/spacecraft/rover/cameras/ (visitado 22-10-2019).

[58] NASA. *NASA Image Galleries*. URL: https://go.nasa.gov/3xIkFh6.

[59] NASA. *Mars exploration image gallery*. 2017. URL: https://www.nasa.gov/mission_pages/mars/images/index.html.

[60] Amos Omondi. *FPGA Implemetations of Neural Networks*. Springer, 2006.

[61] V Pandya, S Areibi y M Moussa. «A Handel-C implementation of the back-propagation algorithm on field programmable gate arrays». En: *2005 International Conference on Reconfigurable Computing and FPGAs (ReConFig'05)*. 2005, 8 pp.-6. DOI: 10.1109/RECONFIG.2005.5.

[62] Vicent L. Pisacane, ed. *Fundamentals of Space Systems*. Oxford university press, 2005. ISBN: 978-0-19.516205-9.

[63] Donald Rapp. *Human Missions to Mars*. Springer, 2008.

[64] David Ratter y Engineer. «FPGAs on Mars». En: *Xcell Journal* (2004).

[65] Donald C. Rizzo. *Fundamentos de amatomia y fisiología*. Tercera. CENGAGE Learning, 2011.

[66] F Rosenblatt. «The perceptron: A Probabilistic Model for Information Storage and Organization in the Brain». En: *Psychological Review* 65 (1958).

[67] M Rossmann, A Bühlmeier, G Manteuffel y K Goser. «Short- and long-term dynamics in a stochastic pulse stream neuron implemented in FPGA». En: 2006, págs. 1241-1246. DOI: 10.1007/BFb0020321.

[68] Carl Sagan. «Blues para un planeta rojo». En: *Cosmos*. Editorial Planeta, 1980. Cap. 5, pág. 106.

[69] V Salapura. «Neural networks using bit stream arithmetic: a space efficient implementation». En: *Proceedings of IEEE International Symposium on Circuits and Systems - ISCAS '94*. Vol. 6. 1994, 475-478 vol.6. DOI: 10.1109/ISCAS.1994.409629.

[70] Valentina Salapura, Michael Gschwind y Oliver Maischberger. «A Fast FPGA Implementation of a General Purpose Neuron». En: vol. 849. 1995. DOI: 10.1007/3-540-58419-6_88.

[71] Minal Sawant. *Past, Present, Future - Xilinx on Mars Rovers...!* URL: https://forums.xilinx.com/t5/Adaptable-Advantage-Blog/Past-Present-Future-Xilinx-on-Mars-Rovers/ba-p/944915 (visitado 22-10-2019).

[72] Aravind Seeni, Bernd Schfer y Gerd Hirzinger. «Robot Mobility Systems for Planetary Surface Exploration State-of-the-Art and Future Outlook: A Literature Survey». En: *Aerospace Technologies Advancements* January 2010 (2010). DOI: 10.5772/6930.

[73] Ghassan Hazin Shakoory. «FPGA Implementation Of Multilayer Perceptron For Speech Recognition». En: *Journal of Engineering and Development* 17, No 6 (2013).

[74] Dee Unglaub Silverthorn. *Human Physiology*. Sexta. Pearson, 2013. ISBN: 13: 978-0-321-75007-5.

[75] K Steinbuch. «Die lernmatrix. Kybernetik (Biological Cybernetics)». En: (1961).

[76] T Szabo, L Antoni, G Horvath y B Feher. «A full-parallel digital implementation for pre-trained NNs». En: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*. Vol. 2. 2000, 49-54 vol.2. DOI: 10.1109/IJCNN.2000.857873.

[77] Ryszard Tadeusiewicz, Rituparna Chaki y Nabendu Chaki. *Exploring neural networks with C sharp*. CRC Press, 2015. ISBN: 13: 978-1-4822-3340-7.

[78] Ronald Tocci, Neal Widmer y Gregory Moss. *Sistemas digitales: principios y aplicaciones*. Décima. Pearson Educación, 2007.

[79] Janet Vertesi. *Seeing Like a Rover*. University of Chicago, 2015. ISBN: 13: 978- 0-226- 15601- 9.

[80] S Warren, Mcculloch y Walter Pitts. «A logical calculus of the ideas immanent in nervous activity». En: *Bulletin of mathematical biophysics* 5 (1943).

[81] Paul j. Werbos. «Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences». PhD thesis. Harvard University, 1974.

[82] Dr. David R. Williams. *Mars Fact Sheet*. URL: https://nssdc.gsfc.nasa.gov/planetary/factsheet/marsfact.html (visitado 13-10-2019).

[83] Peter Wilson. *Design Recipes for FPGAs*. segunda. Elsevier, 2016. ISBN: 978-0-08-097129-2.

[84] Andreas Zell. *Simulation Neuronaler Netze*. Addison-Wesley, 1994.

[85] Jihan Zhu y Peter Sutton. «FPGA implementations of neural networks - A survey of a decade of progress». En: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 2778 (2003), págs. 1062-1066. ISSN: 16113349. DOI: 10.1007/978-3-540-45234-8_120.