UNIVERSIDAD NACIONAL DEL SUR

Tesis Doctor en Ingeniería

CIRCUITOS INTEGRADOS DE ALTO DESEMPEÑO PARA VISIÓN CON PROCESAMIENTO BASADO EN REDES CELULARES

Martín Di Federico

BAHIA BLANCA ARGENTINA

2011

Prefacio

Esta Tesis se presenta como parte de los requisitos para optar al grado Académico de Doctor en Ingeniería, de la Universidad Nacional del Sur y no ha sido presentada previamente para la obtención de otro título en esta Universidad u otra. La misma contiene los resultados obtenidos en investigaciones llevadas a cabo en el Grupo de Investigaciones en Sistemas Electrónicos y Electromecatrónicos, dependiente del Departamento de Ingeniería Eléctrica y de Computadoras durante el período comprendido entre Abril de 2006 y Noviembre de 2010, bajo la dirección de los Profesores Dr. Pedro Marcelo Julián, Profesor Asociado, y el Dr. Favio Masson , Profesor Adjunto, ambos pertenecientes al Departamento de Ingeniería Eléctrica y de Computadoras.

Bahía Blanca, 15 de Marzo de 2011 Martín Di Federico

Departamento de Ingeniería Eléctrica y de Computadoras

Universidad Nacional del Sur

Agradecimientos

Muchas son las personas que colaboraron para que esta tesis se haga realidad. La gente del laboratorio, compañeros de trabajo, compañeros de estudio y de emociones. Gente de esta ciudad y de otras distribuidas por el mundo. Amigos que, con su tiempo, esfuerzo y paciencia supieron acompañarme desde su lugar para que este trabajo sea de frutos. A todos ellos y ellas quiero agradecerles por su colaboración, tiempo, palabras y aliento.

Resumen

En los últimos años, con el surgimiento de sistemas multimedia se ha vuelto popular la incorporación de cámaras con procesamiento en el mismo chip, en productos de consumo como cámaras de video, cámaras fotográficas, teléfonos celulares, reproductores multimedia, etc. En esta tesis se presenta el analisis de una arquitecturas que permite crear una "cámara inteligente" que incorpora la capacidad de procesar las imágenes que adquiere utilizando un procesamiento paralelo distribuido sobre el plano focal. El funcionamiento se basa en una estructura del tipo CNN simplicial, donde cada celda opera en función de su información y la de celdas vecinas. Cada celda implementa una ecuación discreta de evolución de estado, basada en una función lineal por tramos multidimensional. Las celdas se programan a través de una única memoria que se dispone en la periferia del integrado, y el cálculo se realiza con señales codificadas en tiempo, lo cual permite una realización muy eficiente desde el punto de vista del área ocupada por cada celda. Se presentan dos circuitos integrados diseñados bajo estos principios. Se han fabricado dos circuitos integrados, el primero en una tecnología CMOS estándar de $90nm$ que contiene un arreglo de 64×64 celdas. El segundo se diseñó en una tecnología 3D de dos pisos de $0,13\mu m$ y contiene un arreglo de 48×32 celdas.

Abstract

In recent years, with the emergence of Multimedia systems cameras with on-chip processing has become popular, in consumer products like video cameras, cell phones, media players, etc. This thesis presents the analysis of an architecture of a "*smart*" camera that has the ability of acquire and to process images using a parallel processing. This chip works based on a simplicial cnn structure, where each cell operates according to the neighborhood information. Each cell implements a discrete state equation, based on a multidimensional piecewise linear function. The cells are programmed with memory on the periphery of the integrated, and the calculation is performed with time coded signals, which allows very efficient realization in terms of area. Two integrated circuits are presented here, designed under these principles. The first is 64 *times* 64 array fabricated on a 90nm CMOS technology. The second was designed in a 3D 0,13 *mum* technology and contains an array of 48 *times* 32 cells.

UNIVERSIDAD NACIONAL DEL SUR

Fecha: **Marzo 2011**

Autor: **Martín Di Federico**

Titulo: **Circuitos Integrados de alto desempeño para visión con procesamiento basado en redes celulares**

Departmento: **Departamento de Ingeniería Eléctrica y de Computadoras**

Grado: **Doctorado** Mes: **Marzo** Año: **2011**

Por medio de la presente nota se otroga el permiso a la Universidad Nacional del Sur de distrubir y realizar copias sin finalidad comercial de esta tesis a pedido de algún individuo y/o institución.

Firma del Autor

Indice

Lista de Tablas

Lista de Figuras

Capítulo 1

Introducción

1.1. Introducción

Actualmente el procesamiento de imágenes se ha convertido en una herramienta elemental para todas aquellas tareas donde la visión y representación de escenas son esenciales [57, 61, 63]. Debido a que una imagen contiene una gran cantidad de datos e información, su análisis y tratamiento puede llevar un mayor tiempo que el deseado. Por ejemplo: Un filtrado a una imagen de 10 Megapixeles (donde se necesitan realizar 90 Millones de multiplicaciones y 80 Millones de sumas), en un sistema de cómputo corriendo a 2Ghz lleva más de 100ms, pudiéndose procesar menos de 10 imágenes por segundo. Este inconveniente representa una desventaja y un problema para sistemas donde el procesamiento debe ejecutarse en tiempo real o una representación inmediata es necesaria. En el área de visión por computadora la importancia del procesamiento en tiempo real ha ido en aumento ya que se espera en los próximos años poder construir vehículos autónomos, robots de servicios o máquinas que presten asistencia a personas con discapacidades físicas con un desempeño fiable. Una de las tareas de suma importancia al respecto es la detección de objetos, de movimiento y el seguimiento de objetos en tiempo real [61, 63]. En escenas de tráfico es necesario detectar y seguir los objetos que se mueven, los robots de servicio de limpieza de pisos deben de evitar los obstáculos fijos y en movimiento, tal como el hombre o los animales.

Hay varias áreas donde es útil y necesario el procesamiento de imágenes o donde

sistemas de visión son necesarios [58, 64, 59, 60, 65], entre ellos se puede mencionar:

1. Mejoramiento de una imagen digital con fines interpretativos;
2. Toma de decisiones de manera automática de acuerdo al contenido de la imagen digital;
3. Detección de presencia de objetos o reconocimiento del cuerpo humano;
4. Inspección visual automática;
5. Medición de características geométricas y de color de objetos;
6. Clasificación de objetos;
7. Restauración y mejoramiento de la calidad de las imágenes;
8. Desarrollo de sistemas para el diagnóstico y tratamiento médico, aplicando técnicas de visión artificial.
9. Navegación Autónoma
10. Seguimiento de objetos, vehículos o personas.
11. Reconocimiento óptico de caracteres, conocido también como OCR;
12. Identificación de personas (Huellas digitales, venas del ojo);

Para poder obtener un procesamiento en tiempo real se requiere del uso de hardware especializado y computadoras especiales de procesamiento paralelo [54, 57], que realizan tareas de pre-procesamiento como filtrado, segmentación, cálculo de flujo óptico y otras tareas que se explicarán en el Cap. 2. Para realizar esta tarea, se necesitan básicamente dos dispositivos, uno que capte o mida la intensidad de luz (cámara fotográfica), y otro que realice el procesamiento de la información obtenida [56, 62]. En el mercado de circuitos integrados hay dos tecnologías para detectar luz: CMOS (en inglés: Complementary Metal Oxide Semiconductor) [70, 78, 72, 89] y CCD (en inglés: Charged Coupled Device) [51] ; cada una con sus ventajas y

desventajas en función del consumo, ruido, capacidad de tomar fotografías instantáneas, etc. Estas tecnologías evolucionaron mucho entre 1970 y 1990, y la tecnología CMOS tuvo un desarrollo particular como consecuencia de la evolución asociada a los circuitos integrados CMOS. Este crecimiento de complejidad propició el interés en dispositivos de imágenes CMOS. Este interés se vio además acrecentado por el gran volumen de producción en tecnologías estándar CMOS. La tecnología CCD no permite el procesamiento integrado en el mismo chip[36, 22, 29], y además requiere líneas de producción especializadas. Por el contrario, con pocos cambios en los procesos CMOS se llega en la actualidad a la madurez necesaria para implementar las denominadas cámaras inteligentes o cámaras con procesamiento en el mismo chip con la capacidad de tener un gran rango dinamico [52, 69, 74, 83] y de procesar imágenes a una gran velocidad [87, 88, 79]. Un ejemplo de esto son las cámaras fotográficas y de video que realizan auto-foco, reconocen caras, reconocen sonrisas, ajustan la imagen de acuerdo a la intensidad de la luz ambiente, etc. Esta tecnología está siendo cada vez más usada y se puede encontrar integrada en artículos comerciales como cámaras de video, cámaras fotográficas, celulares, etc. En los últimos años, gracias al incremento exponencial de la densidad de integración de transistores, se ha vuelto factible la incorporación de unidades de procesamiento a nivel de pixel en cámaras, que permiten manipular imágenes digitales con el fin de obtener información objetiva de la escena captada por una cámara[1, 49, 25, 50, 19].

Esta capacidad de procesamiento digital de imágenes ha sido ampliamente utilizada por diversas disciplinas tales como:

1. Investigación (satélites, medición automática de parámetros)
2. Medicina (diagnóstico por imágenes, implantes oculares)
3. Seguridad (detección de movimiento, objetos peligrosos, reconocimiento de personas, etc)
4. Industrias (control de calidad, máquinas automáticas)
5. Vida cotidiana (domótica, sistemas automatizados)
6. Arte-Ocio (juegos por computadora, celulares, cámaras fotográficas).

La forma en la cual se manipula la información de una imagen para obtener información también puede variar. Hay varias técnicas, algoritmos y formas de tratar la información que son más eficientes, más veloces o de una implementación más sencilla. Las Redes Neuronales Celulares (CNN) bidimensionales constituyen una atractiva arquitectura sobre la que se pueden implementar varios procesos para extraer información o realizar una interpretación de la imagen.

Las CNN se definen como un arreglo de procesadores no lineales localmente interconectados (llamados células o celdas) que operan en paralelo. Las CNN presentan importantes ventajas como su inherente robustez y control sencillo de la estructura, que se define por una función externa y es la misma para todas las celdas. En esta tesis se plantea una arquitectura basada en una generalización de la red CNN denominada CNN simplicial o S-CNN, que funciona en tiempo discreto de manera digital. Esta red, a diferencia de la propuesta por Chua como CNN estándar -utilizada como plataforma de desarrollo por la gran mayoría de los integrados basados en CNN - utiliza una función no lineal para definir el sistema dinámico de la celda. Más específicamente, la función utilizada es una función lineal por tramos (PWL por sus siglas en inglés "piecewise linear"), que puede ser programada mediante valores almacenados en una memoria. Esto le confiere mayor generalidad y versatilidad en comparación con la CNN estándar. La arquitectura planteada es completamente digital, con lo cual hereda las propiedades de escalado en área y velocidad de los procesos digitales CMOS. Se han estudiado varios algoritmos de análisis de datos y de procesamiento de imágenes, para lograr crear una celda con gran capacidad de procesamiento. A la red celular se le han agregado estructuras para mejorar y aumentar la capacidad de procesamiento global. Como consecuencia del análisis de algoritmos y de estructuras de cálculo, se han producido varios circuitos integrados experimentales.

El primer circuito integrado de prueba se diseñó sobre un proceso CMOS estándar de 0,35μm. Se diseñaron dos circuitos distintos como prueba de concepto, uno utilizó lógica dinámica de precarga y evaluación para la realización de las celdas, de manera de obtener una mayor densidad de pixeles, y el otro lógica estándar CMOS. En este primer circuito de prueba solo se implementó una celda aislada con conexiones al exterior, de manera poder testear su funcionamiento previo a su inclusión

en un flujo de diseño automatizado. Al mismo tiempo que se realizó el diseño del circuito integrado mencionado, se llevó a cabo un procedimiento de testeo sobre un circuito integrado previamente diseñado por miembros del Grupo de Investigación en Sistemas Electrónicos y Electromecatrónicos (GISEE) de la UNS. El circuito integrado implementa un arreglo hexagonal de 7×6 elementos en una tecnología de $0,5\mu m$, basado en lógica dinámica de precarga y evaluación.

El segundo circuito integrado se diseñó sobre un proceso de $90nm$ de la empresa UMC. Este circuito fue completamente diseñado simulado y dibujado a mano. Las celdas están colocadas en un arreglo ortogonal de 64×64 pixeles, y cada celda está comunicada con sus ocho vecinos a través de dos conjuntos de conexiones configurables. Este diseño es de $2mm \times 2mm$ y tiene más de 2.300.000 transistores. El tercer circuito se diseñó y fabricó en una novedosa tecnología de integración llamada 3D, que permite interconectar de manera muy eficiente varias obleas de silicio sobre aislante (SOI) de $0,13\mu m$. En este caso, todo el sistema fue descripto en VHDL y se utilizó un software de diseño por computadora para la colocación y la interconexión de las celdas estándares. Se optó por una configuración rectangular donde cada celda tiene conectividad con ocho celdas vecinas. El prototipo fabricado ocupa un área de $2mm \times 2,5mm$ y contiene un arreglo de 48×32 celdas.

1.2. Organización de la tesis

La tesis está organizada de la siguiente manera: En el Capítulo 2, se presentan los sistemas de visión en general. Se describen algunas de las técnicas de procesamiento de imágenes, se muestran algunos algoritmos de procesamiento de bajo, medio y alto nivel. Se muestran ejemplos de filtrado, segmentación, operaciones morfológicas, localización orientación, obtención de descriptores y como es el proceso de extracción de información de imágenes. Luego se mencionan algunos resultados existentes de chips de visión. Todo este análisis, se realizó para evaluar qué tipo de estructuras se deben colocar en un chip que realice procesamiento y análisis de imágenes. En el Capítulo 3, se introducen las redes celulares no lineales, su descripción matemática y dos casos particulares la CNN estándar y la S-CNN. Se explica con detalle el funcionamiento de la S-CNN, especificando la secuencia de pasos que debe seguir

un circuito integrado con esta metodología de cálculo. En el Capítulo 4 se describe la arquitectura de un sistema de procesamiento S-CNN, y los bloques necesarios para realizar su algoritmo detallándose las tareas que debe realizar cada uno de los bloques, y las diferentes formas de llevarlas a cabo. Se analizan varias arquitecturas, describiendo sus características principales y su impacto en el desempeño del circuito, considerando algunas variantes de la arquitectura y los efectos que tiene en la performance de la celda. En los Capítulos 5 y 6, se describe la realización, desde su diseño hasta la fabricación, de dos chips utilizando diferentes tecnologías y diferentes flujos de diseño. El primero, en tecnología CMOS de $90nm$ diseñado con un flujo Bottom-Up, y el segundo en una tecnología de integración 3D de $130nm$ diseñado con un flujo de diseño Top-Down. En ambos capítulos, Se muestran algunos de los aspectos más importantes de la tecnología, se describe paso a paso el flujo de diseño seguido, incluyendo el tipo de arquitectura seleccionada, los programas utilizados para desarrollar, sintetizar, simular, crear las máscaras y testear el diseño. Se explican detalles de cada una de las celdas del arreglo y las estructuras extras que lo componen. Se detallan las señales que controlan el circuito, sus funciones y su evolución en el tiempo para realizar algunas tareas básicas. Se muestran valores de simulaciones y testeos realizados. En el Capítulo 7 se describen los sistemas de verificación que se crearon para realizar las mediciones a los circuitos integrados fabricados. Se introducen las variables más importantes en las mediciones de imagers con procesamiento CNN y se muestran resultados de las mediciones de varios circuitos integrados fabricados.

Capítulo 2

Conceptos preliminares y resultados existentes

2.1. Introducción

La visión artificial es la rama de la ciencia y la tecnología que estudia las máquinas con capacidad de visión. La misión principal de un sistema de visión artificial es entender una imagen y sus características [53, 55, 56]. Hay numerosas ramas de la ciencia que tienen participación en esta área, como la óptica, el procesamiento de señales, la estadística, la geometría, la inteligencia artificial, etc. Para poder "*entender*" una imagen hay que obtener información de ella y luego analizarla para obtener algún resultado. Para esto se utilizan técnicas de procesamiento y análisis de señales las cuales, como se muestra en la Fig. 2.1, se pueden separar en varios pasos.

El primer paso radica en la obtención y digitalización de la imagen. Para esto se utiliza un dispositivo transductor de luz, que convierte la intensidad de luz en una variable eléctrica, la cual con un conversor analógico digital (A/D) se convierte a una imagen digital []. Luego una etapa de pre-procesamiento o de procesamiento de bajo nivel, se encarga de preparar la imagen para su posterior análisis. Las operaciones típicas en esta etapa son la disminución de ruido a través de filtros espaciales, realce de regiones, de contraste, y las operaciones que se denominan de *bajo nivel*. Como siguiente paso se realiza el análisis de la imagen, cuyo fin es la obtención de

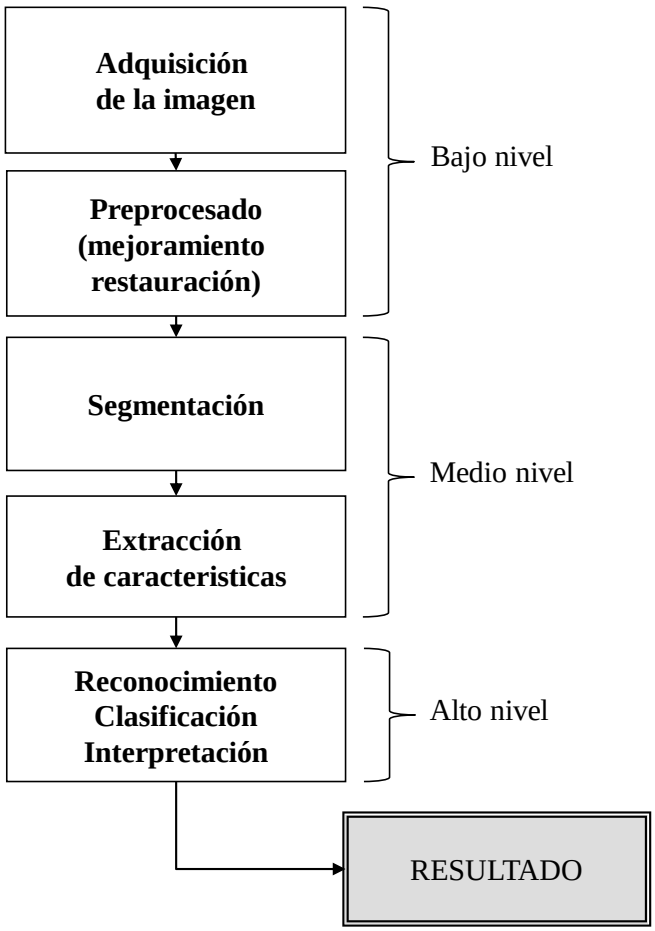


Figura 2.1: Esquema de extracción de información de imágenes.

información contenida en la misma. Esta tarea se realiza descomponiendo la imagen en partes o unidades que comparten alguna característica. Estas características son altamente dependientes de la aplicación del sistema de visión; por ejemplo para reconocer personas o huellas dactilares, las zonas de interés y las características que comparten son diferentes. Las operaciones típicas en esta etapa son la segmentación, la obtención de descriptores, y las operaciones que se denominan de *nivel medio*. Las salidas generadas en este paso son imágenes muy simples (muchas veces binarias) acompañadas de datos que representan o describan la imagen. Por último se encuentra la etapa que se encarga de etiquetar, clasificar y analizar la información para realizar el reconocimiento o interpretación. Aquí entra en juego el reconocimiento de

objetos, que es el encargado de identificar y detectar objetos analizando sus características. Las operaciones típicas en esta etapa son la correlación de la información, análisis de descriptores, y las operaciones que se denominan de *alto nivel*. El análisis de la información obtenida, ya sean estadísticos, de regresión, o con redes neuronales es utilizado para tomar algún tipo de decisión, o entregar algún resultado que puede servir para mejorar el procesamiento y el análisis de imágenes o para producir algún tipo de aprendizaje en sistemas de mayor complejidad.

En algunas aplicaciones de visión, especialmente donde se requiere un alto poder computacional junto con bajo consumo, es beneficioso el uso de "chips de visión dedicados" (dispositivos que combinan captura de imágenes y procesamiento en la misma pastilla de silicio). Los chips de visión de aplicación específica pueden realizar eficientemente algunas tareas poco complicadas o de bajo nivel, como filtrado, detección de bordes o detección de movimiento. Algunos de estos circuitos realizan el procesamiento de manera analógica otros de manera digital, algunos procesan la información de manera secuencial y otros trabajan en paralelo. Algunos cuentan con una estructura de software puede para combinar diferentes operaciones, ejecutar algoritmos complejos y proveer mayor flexibilidad. Va le destacar que la selección de la arquitectura del sistema y la forma de procesar la información depende de la aplicación para la cual se use el circuito.

En este capítulo se describen algunos de los procesos y algoritmos involucrados en el procesamiento y análisis de imágenes. En la Sección 2.2 se explican algunas técnicas y algoritmos para realizar procesamiento de imágenes binarias, mejoramiento de la imagen, segmentación, mediciones en imágenes, obtención de descriptores, cálculo de flujo óptico, codificación en código cadena y reconocimiento de objetos. Estos conceptos servirán luego para diseñar la arquitectura de un chip que tenga la posibilidad de entender, o extraer información de una imagen. En la Sección 2.3, se describen los resultados existentes en la literatura sobre cámaras con capacidad de procesamiento. Se muestran varios tipos de procesadores del tipo paralelo considerando los del tipo CNN, digital y analógico, y también procesadores más genéricos del tipo SIMD (Single Instruction Multiple Data).

2.1.1. Características de imágenes digitales.

Una imagen digital consta de un arreglo de celdas llamadas píxeles, ordenadas en una cuadrícula. Cada uno de ellos guarda la información de la intensidad de luz o brillo de la imagen en la región comprendida por el pixel. En una imagen de escala de grises, el valor de brillo de cada punto o pixel, se almacena en un byte o registro. La cantidad de bits del registro, define el rango de la escala de grises, o cantidad posible de tonos que puede tomar la imagen; por ejemplo, si el registro es de 8 bits, los tonos pueden variar entre 255 (Blanco) y 0 (Negro). Si se desea tener un valor más detallado del nivel de brillo del pixel, se necesita codificar la imagen con una mayor cantidad de valores (mayor cantidad de bits). Si la imagen es a color, la precisión de los registros donde se almacena la información define la profundidad de color. La cantidad de píxeles por unidad de longitud define la resolución de la imagen. Normalmente se mide en "pixeles por pulgada"(ppp o dpi) o directamente indicando la cantidad de pixeles totales en las filas y las columnas (640×480).

Cada pixel posee un área finita y abarcará una región determinada en la imagen determinando la frecuencia espacial que se puede muestrear, o el tamaño mínimo de una línea que se puede captar. Al aumentar la resolución se pueden obtener imágenes con una mayor frecuencia espacial, lo que permite capturar líneas más finas de la imagen, o sea, mayor cantidad de detalles. La Fig. 2.2 muestra una imagen variando la resolución y Fig. 2.3 muestra la variación de la misma imagen variando la precisión de los registros.

2.2. Procesamiento y análisis de imágenes

El procesamiento de imágenes es el conjunto de técnicas y algoritmos que se aplican a imágenes con el objetivo de mejorar su calidad o para el proceso de extraer información. El análisis de imágenes es el conjunto de tareas que tienen como fin obtener información. En esta sección se explican algunas técnicas y algoritmos de procesamiento y análisis de imágenes en varios niveles de abstracción.

Las técnicas de procesamiento y análisis de imágenes se pueden dividir en tres grupos básicos:

Figura 2.2: Variaciones de una imagen al variar la cantidad de pixeles(512×512, 256×256, 128×128, 64×64).

- Procesamiento de bajo nivel: La función de estos procesos es la de mejorar la imagen para fines interpretativos y prepararla para obtener información. La salida de estas funciones son otras imágenes a las cuales se les ha hecho alguna modificación como disminución del ruido, filtrado, amplificación de bordes, etc. Algunas de las tareas básicas que se pueden agrupar dentro de este nivel pueden ser: filtrado espacial, filtrado por transformada, resta de imágenes, multiplicación y división de imágenes, enmascaramiento, detección de bordes, etc.

- Procesado de medio nivel: El proceso de nivel medio se refiere a la labor de extracción y caracterización de los componentes de la imagen que se obtienen de un proceso de bajo nivel. La entrada de estos procesos son imágenes y

Figura 2.3: Variaciones de una imagen al variar la cantidad de bits con la que se representa (1, 2, 3, 4, 6 y 8 bits).

las salidas pueden ser otras imágenes o valores numéricos obtenidos de la imagen. Este proceso abarca; la segmentación que es el proceso de separar la imagen en sus partes básicas o en partes que compartan alguna propiedad o característica, la generación de descriptores que son valores numéricos que caracterizan o definen la forma de un objeto, la detección de movimiento que la salida puede ser otra imagen en la que solo se ve el objeto que se movió, o un valor numérico indicando si hay movimiento o que porcentaje de la imagen vario, etc.

- Procesado de alto nivel: En este grupo se encuentran las funciones que se encargan de reconocer o interpretar información obtenida de la imagen. Algunas de las tareas que se realizan en esta área incluyen el reconocimiento de patrones u objetos, la clasificación, el seguimiento, etc. La mayoría de las funciones de nivel medio y bajo utilizan un conjunto de formulaciones matemáticas bien definidas, sin embargo el reconocimiento y la interpretación es un proceso más

especulativo o estadístico.

2.2.1. Mejoramiento de imágenes

El mejoramiento de imágenes se utiliza tanto para corregir los defectos en imágenes adquiridas o para mejorar su aspecto global. Se utilizan en general métodos que incrementan la visibilidad de una porción, aspecto o componente de la imagen, generalmente a expensas de otro, cuya visibilidad es disminuida. El mejoramiento de imágenes se puede realizar en el dominio espacial o en otros dominios como el frecuencial. En el dominio espacial los valores de los pixeles pueden ser modificados de acuerdo a reglas que dependen del valor original del pixel y también de su vecindad, también pueden ser combinados o comparados con otros en su proximidad en una gran variedad de formas. Hay varias técnicas que se pueden utilizar para realizar el mejoramiento de imágenes, como filtrados por convolución o ecualización de histograma.

Filtrado espacial lineal / no Lineal

El filtrado es un tipo de operación que altera el valor de un píxel en función de los valores de los píxeles que lo rodean, es por ello que a este tipo de procesamiento también se lo denomina procesamiento basado en la vecindad u operación de vecindad. El término "*espacial*" se utiliza para distinguir que la alteración del píxel se realiza dependiendo de los valores de los píxeles del entorno sin realizar ninguna modificación previa de sus valores, lo cual no ocurre con el filtrado frecuencial que requiere de la aplicación de la transformada de Fourier. Si la función utilizada para alterar el valor del píxel depende en forma lineal de los píxeles del entorno entonces el filtro es lineal, caso contrario, el filtro es no lineal. La Fig. 2.4 muestra una imagen filtrada con un filtro de mediana (no lineal), y la Fig. 2.5 filtrada por un filtro promediador (lineal).

Un caso particular de filtros lineales es el de los filtros de convolución cuya operación matemática se define como:

$$f(x) * h(x) = \int_{-\infty}^{\infty} f(x)h(u-x)dx \tag{2.1}$$

Figura 2.4: Filtro no lineal: a)Imagen original c) Imagen filtrada.

Estos filtros se aplican realizando la convolución de la imagen con una máscara o kernel de tamaño variable. , donde la convolución bidimensional discreta de una imagen I por una máscara cuadrada h de lado n+1 se define como:

$$I'(x,y) = \frac{1}{D}\sum_{i=0}^{n}\sum_{j=0}^{n} I(i,j)\cdot h(x-i,y-j) \tag{2.2}$$

La imagen resultante depende de la forma y los valores del kernel seleccionado, pudiéndose realizar básicamente dos tipos de filtros, pasa bajo y pasa alto. La Fig. 2.5 muestra una imagen a la que se le aplicó un filtrado de convolución, donde a) es la imagen original b) es la máscara utilizada para realizar la convolución y c) es la imagen resultante luego de aplicar la convolución de la imagen con la máscara.

Ecualización de Histograma

El histograma de una imagen indica cuantos píxeles se encuentran en la imagen para cada valor posible de brillo. Esto se grafica mediante un diagrama donde el eje X indica los valores posibles de intensidad (la escala de grises) y el eje Y la cantidad de pixeles que poseen ese brillo. La ecualización del histograma pretende modificar la intensidad de los píxeles para lograr un histograma distinto, que cumpla con alguna característica deseada, ya sea ampliar el rango dinámico, corregir algún intervalo, realizar una transformación logarítmica o gamma (para mejorar el contraste de una

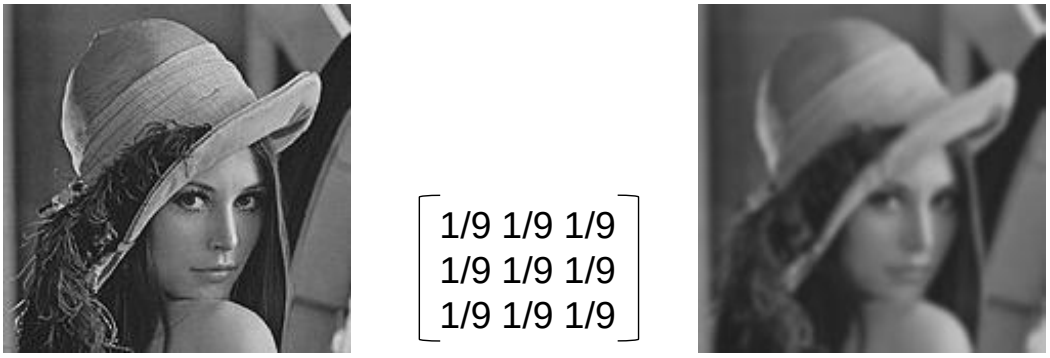


Figura 2.5: Filtro Lineal: a)Imagen original b) Filtro mediana (kernel) c) Imagen filtrada.

imagen), obtener una imagen con una distribución uniforme del histograma (mejora el brillo de la imagen), etc. La Fig. 2.6 muestra una imagen (a) con su histograma (b) a la cuál se le aplicó la transormación (e) quedando la imagen ecualizada (c) y su histograma (d).

2.2.2. Procesamiento de imagines Binarias

Las imágenes binarias consisten en grupos de pixeles donde su valor puede ser "0" ó "1", asignados a los colores negro y blanco. Estos valores pueden pertenecer a imágenes que fueron obtenidas directamente con esa precisión o fueron determinados por un proceso de segmentación de una imagen más compleja. La mayoría de las funciones que se pueden realizar con imágenes binarias se pueden separar en dos categorías: operaciones booleanos para combinar imágenes y operaciones morfológicas que modifican pixeles individuales de una imagen en particular.

Operaciones booleanas

Las operaciones booleanas toman el valor del pixel como un valor lógico, pudiéndole aplicar las funciones logias conocidas, AND ($\cdot$), OR ($+$), XOR ($\oplus$), NOT ($\overline{A}$), etc. La Fig. 2.7 muestra algunas operaciones. Estas operaciones se pueden utilizar para enmascarar imágenes, para unir imágenes o para calcular diferencias entre

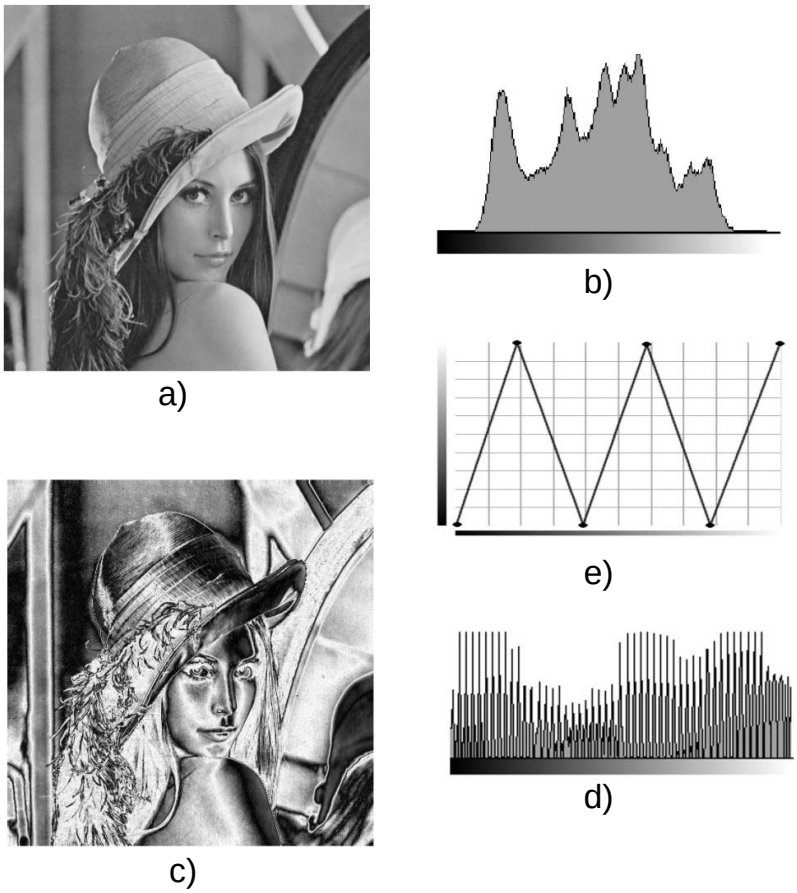


Figura 2.6: Ecualización de histograma: a,b) imagen original y su histograma c,d)imagen ecualizada y su histograma e)función transformación utilizada.

imágenes.

Operaciones Morfológicas

En términos generales, la palabra morfología se refiere al estudio de forma y estructura. En una operación morfológica, el valor de cada píxel en la imagen de salida depende del valor de ese píxel en la imagen de entrada y de su relación con la vecindad, seleccionando el tamaño y forma de la vecindad, a través de un operador morfológico. Los operadores morfológicos están formados por una forma de referencia, llamada elemento estructural, el cual es comparado con la forma original, (imagen) y un mecanismo que detalla cómo debe ser la comparación. Básicamente,

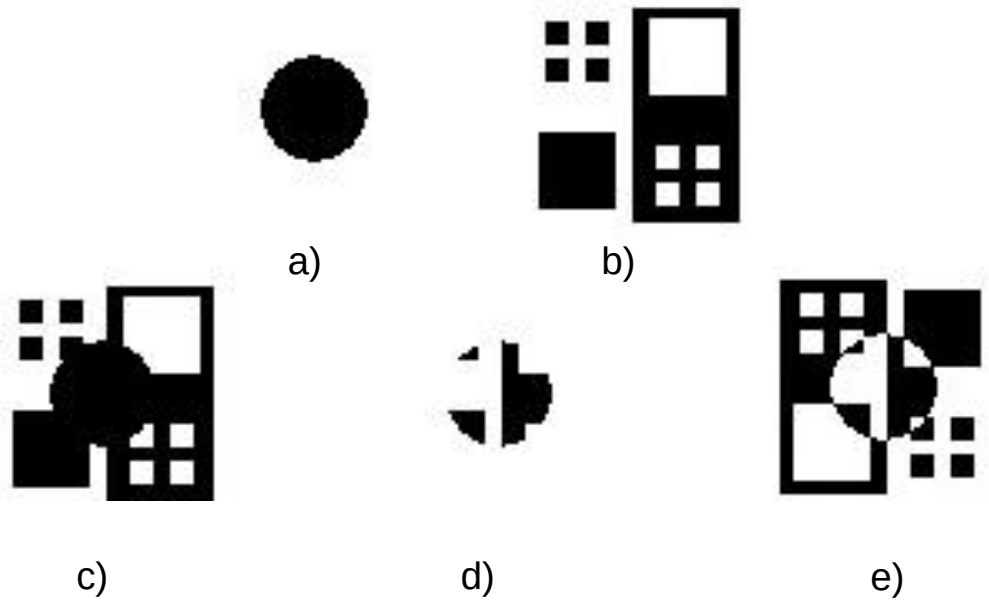


Figura 2.7: Operaciones booleanas con imágenes. a)Imagen A, b)Imagen B, c)$(A \cdot B)$, d)$A + B$, e)$A \oplus B$

en una operación morfológica se modifican los pixeles de la imagen en función de la comparación con el elemento estructural. El resultado de una operación morfológica es otra imagen del mismo tamaño. Las operaciones morfológicas pueden emplearse con variados fines, como detección de bordes, segmentación o realce de algunas características. En base a las operaciones morfológicas, se pueden construir filtros para utilizar en lugar de los filtros lineales estándar. Cabe destacar que los filtros lineales suelen distorsionar la forma geométrica de la imagen, a diferencia de los morfológicos que, mayormente, la dejan intacta. A continuación se detallan algunas de las operaciones morfológicas más usuales.

- Erosión y dilatación.

  La operación erosión ($\ominus$) tiene como fin reducir los píxeles en los bordes de un objeto. El elemento estructural (EE) usualmente es un arreglo de 3×3, pero puede tomar cualquier forma, y el valor del pixel central, o destino, es el menor de todos los que se encuentren dentro del EE. El número de píxeles a los que se aumenta o reduce el nivel depende del tamaño y forma del elemento estructural usado para procesar la imagen. La erosión contrae la

imagen original (suponiendo que la imagen es blanca, y el fondo es negro). La erosión en una imagen X con un elemento estructural B se define como,

$$X \ominus B = min(x|x \in X \bigcap B) \tag{2.3}$$

Esta operación tiende a adelgazar las zonas blancas y engrosar las zonas negras, tal como se puede ver en la Fig 2.8

Figura 2.8: Erosión de una imagen.

La operación dilatación ($\oplus$) tiene como fin aumentar los píxeles en los bordes de los objetos presentes en la imagen. Para calcular la dilatación se superpone el píxel central del EE con cada píxel de la imagen de entrada, y el píxel destino será el máximo entre todos los píxeles presentes en el EE. La dilatación expande la imagen original (suponiendo que la imagen es blanca, y el fondo es negro)

$$X \oplus B = max(x|x \in X \bigcap B) \tag{2.4}$$

Esta operación tiende a adelgazar las zonas negras y engrosar las zonas blancas, tal como se puede ver en la Fig 2.9

Figura 2.9: Dilatación de una imagen.

- Apertura y cierre

  Las operaciones de erosión y dilatación se combinan para formar diferentes métodos de procesamiento. Uno de ellos es la apertura ($\circ$) de una imagen.

Esta se define como la realización de una erosión seguida de una dilatación utilizando el mismo elemento estructural en ambas operaciones. Este método se aplica para eliminar pequeños objetos y mantener el tamaño en los grandes (eliminar ruido). La Fig. 2.10 muestra una imagen y su apertura, donde se puede observar que han desaparecido las puntas negras y se han redondeado.

$$X \circ B = (X \ominus B) \oplus B \tag{2.5}$$

Figura 2.10: Apertura de una imagen.

La operación de cierre ($\bullet$) se realiza aplicando la dilatación y luego la erosión (contrario a la apertura). Este proceso se caracteriza por rellenar huecos y conectar objetos que están próximos entre sí. La Fig. 2.11 muestra una imagen y su cierre, donde se puede observar que han desaparecido las puntas blancas y se han redondeado.

$$X \bullet B = (X \oplus B) \ominus B \tag{2.6}$$

Figura 2.11: Cierre de una imagen.

- Transformación acierta o falla (AOF) $\circledast$ La transformación de acierta o falla (AOF) se utiliza para localizar cierta forma definida (B) en una imagen (A). Se compara la forma del elemento a comparar con la imagen y si ambas concuerdan, el pixel queda activado. La función de AOF se define como:

$$X \circledast B = (X \ominus B_1) \bigcap (X^c \ominus B_2) = (X \ominus B_1) - (X^c \oplus B_2) \tag{2.7}$$

Siendo la Imagen A y el objeto a localizar B=(B1, B2), con B1 : Elementos que pertenecen al objeto y B2 elementos que pertenecen al fondo.

- Transformaciones adelgazamiento y ensanchado
  La función de adelgazamiento ($\triangleright$) se utiliza para eliminar partes definidas de un objeto. La forma y la ubicación del objeto a sacar de la imagen se obtiene con una transformación AOF, siendo la función adelgazamiento, la resta de la imagen original y el resultado de la operación de AOF.

$$X \triangleright B = X - (X \circledast B) \tag{2.8}$$

  El ensanchado ($\triangleleft$) es una transformación morfológica que se utiliza para hacer "*crecer*" regiones seleccionadas o definidas. Al igual que el adelgazamiento el objeto a "*agregar*" se obtiene con una transformación AOF, y la función en este caso es la suma o unión de la imagen original y el resultado de la operación de AOF.

$$X \triangleleft B = X \bigcup (X \circledast B) \tag{2.9}$$

- Esqueletonización y líneas de vecindad
  El esqueleto de una imagen o un objeto, es una versión más delgada de su forma, equidistante a sus bordes. El esqueleto generalmente hereda las propiedades geométricas y topológicas de un objeto. Esta representación contiene toda la información para conocer y reconstruir la forma del objeto. Las líneas de vecindad de una imagen son las líneas que separan una parte de otra, es decir sus bordes. Esta representación también contiene toda la información de la forma de la imagen. La Fig. 2.12 muestra una imagen con su esqueleto y sus líneas de vecindad. Para obtener el esqueleto de una imagen, se puede aplicar la erosión iterativa que remueve los pixeles de los bordes externos, asegurándose que no se produzca una discontinuidad en la imagen (que no se separe en 2 o más imágenes). Las líneas de vecindad se pueden obtener con un algoritmo de detección de bordes.

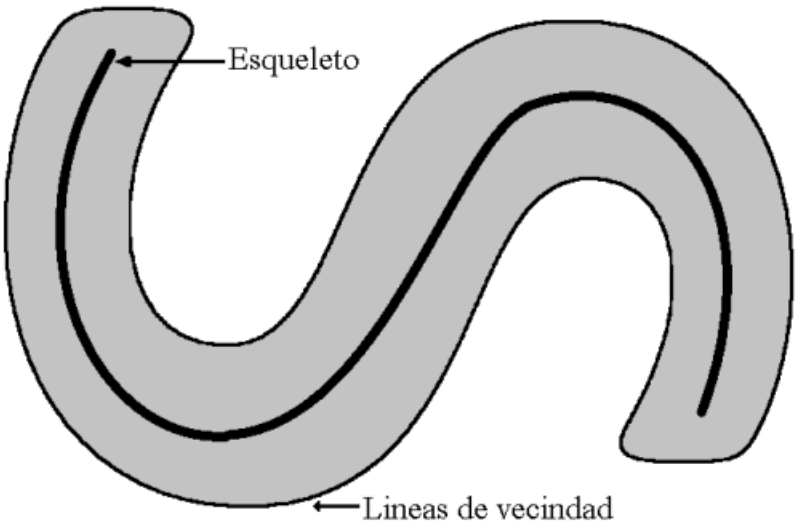


Figura 2.12: Esqueleto de una imagen

- Otras transformaciones

  Envoltura convexa: Consiste en la unión iterativa de la transformación AOF hasta que no se produzcan más cambios, para ninguno de los elementos seleccionados en la imagen. Seleccionando los elementos a buscar como elementos cóncavos, al finalizar la iteración no habrá en la imagen ningún elemento cóncavo, y el resultado será la cobertura convexa de la imagen original.

  Rellenado de regiones: Esta transformada consiste en rellenar las partes internas de las regiones. Comenzando con una imagen blanca (negra) e ir erosionándola (dilatándola) mientras no encuentre ningún objeto.

  Granulación: Esta transformada consiste en erosionar la imagen hasta que quede un solo punto por cada región, pudiéndose tomar este como el centro de gravedad.

Bordes o Gradiente: Consiste en la obtención de los bordes de la figura, restándole a la imagen dilatada la imagen erosionada. Se pueden realizar también variaciones como restar la imagen erosionada a la original, o restar la imagen original a la dilatada.

2.2.3. Segmentación

La segmentación es el proceso de dividir una imagen en varias partes o regiones que correspondan a unidades estructurales de la escena u objetos de interés. El objetivo principal de la segmentación es simplificar la representación de una imagen de una forma simple y fácil de analizar. Al segmentar la imagen, se le asigna a cada pixel una "*etiqueta*" indicando a que grupo, conjunto o parte de la imagen pertenece. Sabiendo que los pixeles con la misma etiqueta comparten alguna propiedad. Una forma de segmentación es la separación de objeto y fondo, donde se selecciona una parte de la imagen (el objeto) y se descarta el resto (el fondo). La segmentación de imágenes es un paso muy importante en el análisis de imágenes, dado que las regiones en las que se separa la imagen serán las que se analicen y, si la segmentación no es adecuada, los niveles superiores no contaran con información confiable. Hay muchas formas diferentes de segmentar imágenes, en el sitio http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/ se encuentra *The Berkeley Segmentation Dataset and Benchmark* que es una base de datos de figuras segmentadas por humanos, que se utiliza como patrón para realizar comparaciones con algoritmos. La Fig. 2.13 muestra un ejemplo de una imagen y las diferentes formas en las que se la puede segmentar.

Dentro de las formas de realizar segmentaciones se encuentran las siguientes:

- Segmentación por histograma: Se utiliza la información del histograma para realizar la segmentación. Cada elevación en el histograma, o moda, está asociada a una determinada región de la imagen. Los valles del histograma pueden ser utilizados como posibles umbrales para separar esas regiones. La idea principal radica en que los píxeles con valores de niveles de grises por debajo del umbral se asocian a una región y el resto a otra. Si se separa en

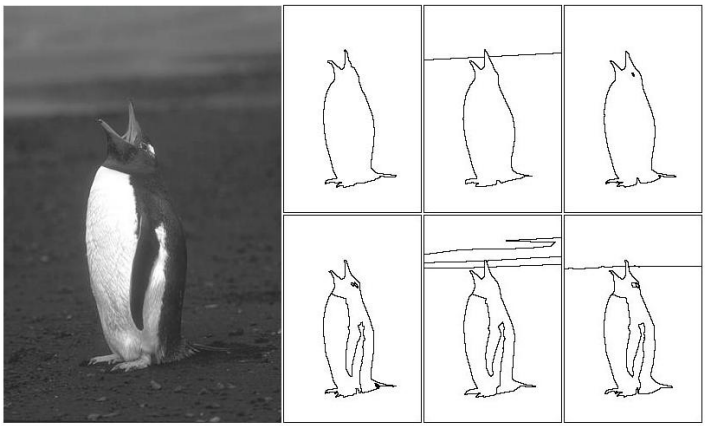


Figura 2.13: Segmentación.

dos grupos la acción se llama umbralizado o binarización. Si se crean más de dos grupos, el procedimiento se llama obtención de Curvas de nivel. Cuando se utiliza la información de toda la imagen en su conjunto para obtener un umbral, el método se denomina umbralización global, que puede ser múltiple si se utiliza más de un umbral. Si el método se aplica en una vecindad de un píxel se dice que la umbralización es local. La Fig. 2.14 muestra una imagen segmentada.

Figura 2.14: Segmentación de una imagen a)Imagen Original b) Método de histograma c) Separado objeto de fondo de manera manual (deseado).

- Clustering: Este método radica en separar los pixeles en grupos o clases, definidas (ya sea por distancias, posiciones, brillo, u otras propiedades). Entre los métodos más comunes se encuentra el K-media donde se separa la región en

K clases y se minimiza la distancia total. En el método de variación local, se crean grupos donde los pixeles del mismo grupo tienen poca variación local, es decir son parecidos o tienen poca dispersión de brillo.

- Segmentación basada en bordes: Los bordes se pueden utilizar como delimitadores de regiones. Dada una imagen, se le aplica un filtro pasa alto para reconocer los bordes, y luego por métodos de crecimiento de bordes, o de relleno de los interiores, se separa la imagen en regiones.

- Segmentación utilizando grafos: Para este método, la imagen se modela como un grafo ponderado, asignando un grupo de pixeles a cada nodo, y la similitud entre los pixeles (nodos) es el peso de las aristas. Luego se agrupan los nodos en función de los pesos. Algunos algoritmos en este área son arboles de expansión mínima, corte normalizado, mínimo corte, etc.

- Segmentación basada en regiones: Este método de segmentación se puede realizar de maneras diferentes. En el método llamado *Crecimiento de regiones*, se "*siembran*" semillas que definen una zona, y se hacen crecer (dilatación), siguiendo una regla establecida (dependiendo del brillo, del gradiente, etc.). Luego de varias iteraciones, las semillas cubren toda la imagen separándola en regiones. Son posibles también otros dos enfoques que prescinden de la utilización de semillas iniciales: uno ascendente y otro descendente. El primero consiste en partir de un grupo de regiones que cubran toda la imagen y unir aquellas que posean fronteras comunes, con la condición de que la nueva región conserve las mismas cualidades que las regiones originales. El segundo enfoque parte de la imagen como un todo y prosigue con la división sucesiva de aquellas regiones que no sean homogéneas mientras exista alguna. También es posible utilizar una estrategia híbrida, es decir de división y fusión.

- Watershed: Una imagen en escala de grises puede ser vista como un relieve topográfico, donde la escala de grises indica la altura. Si suponemos que los objetos de interés se encuentren a diferentes alturas y están delimitados por elevaciones más altas que las elevaciones internas, se pueden utilizar los

"*valles*" (mínimos locales) como semillas e ir agrupando los pixeles por la altura. Este algoritmo es un tipo de crecimiento de regiones, donde se comienza con los valles, y se simula una "*inundacin*". La diferencia con una umbralización radica en que las áreas que quedan no poseen huecos. Otra forma de realizar este procedimiento es a través de erosiones sucesivas.

2.2.4. Mediciones en imágenes

El resultado de la medición de una característica o variable en una imagen indica cuantitativamente alguna propiedad de la imagen o del objeto. Los valores obtenidos de las mediciones de imágenes se pueden utilizar para realizar un análisis estadístico y así reconocer objetos, hacer un seguimiento, etc. Existe una gran cantidad de variables o valores que se pueden medir en una imagen como brillo, locación, tamaño, forma, etc. A continuación se muestran algunas de ellas.

Brillo

Como se ha visto anteriormente, el valor numérico que cada pixel guarda, es a menudo, el brillo de la imagen en ese punto y dependiendo de la precisión del píxel varia la cantidad de valores que se pueden almacenar. El histograma de una imagen indica cuantos pixeles se encuentran en la imagen para cada valor posible de brillo. Con estas mediciones de brillo, se puede obtener información del contraste de una imagen, conocer su rango dinámico y se puede obtener valores como, la media (valor promedio de la imagen), la moda (valor que aparece con mayor frecuencia), la mediana (punto medio de los valores ordenados de menor a mayor), etc. La Fig. 2.15 muestra el histograma de una imagen y algunos de los valores que se pueden obtener.

Tamaño y ubicación

Para obtener información acerca de la ubicación y el tamaño de un objeto en una imagen, se necesita conocer la cantidad de pixeles que se encuentran en el objeto a medir. Este es uno de los procedimientos más comunes usado en el procesamiento de imágenes, que permite obtener una gran cantidad de información.

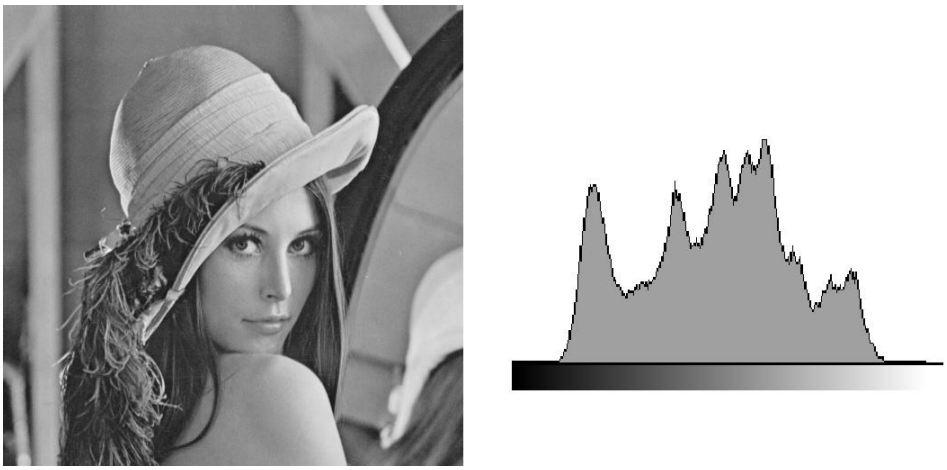


Figura 2.15: Una imagen en escala de grises y su histograma.

Entre las medidas de longitud se puede nombrar el área, la cual es una medida de distancia al cuadrado, y la longitud que es una medida de distancia lineal.

- Mediciones de área: La medida básica de tamaño de un objeto es el área. Desde el punto de vista de una imagen, el área es la cantidad de pixeles que se encuentran dentro del objeto y esta se obtiene directamente contando los pixeles. Como se muestra en la Fig. 2.16, se pueden tener en cuenta varias mediciones posibles de área, para tener un valor aproximado del objeto real.

    - Área neta: Área de la imagen o del objeto.
    - Área rellena: Área de la imagen, contando también los huecos de la misma.
    - Área Convexa: Área de la figura convexa más pequeña que cubre la imagen.
    - Área del círculo que lo contiene: Área del círculo más pequeña que cubre toda la imagen.
    - Área del rectángulo que lo contiene: Área del rectángulo más pequeño, con cualquier inclinación que cubre la imagen por completo.

- Mediciones de longitud: También se pueden realizar medidas cuya dimensión es de distancia. A continuación se muestran algunas mediciones que se pueden

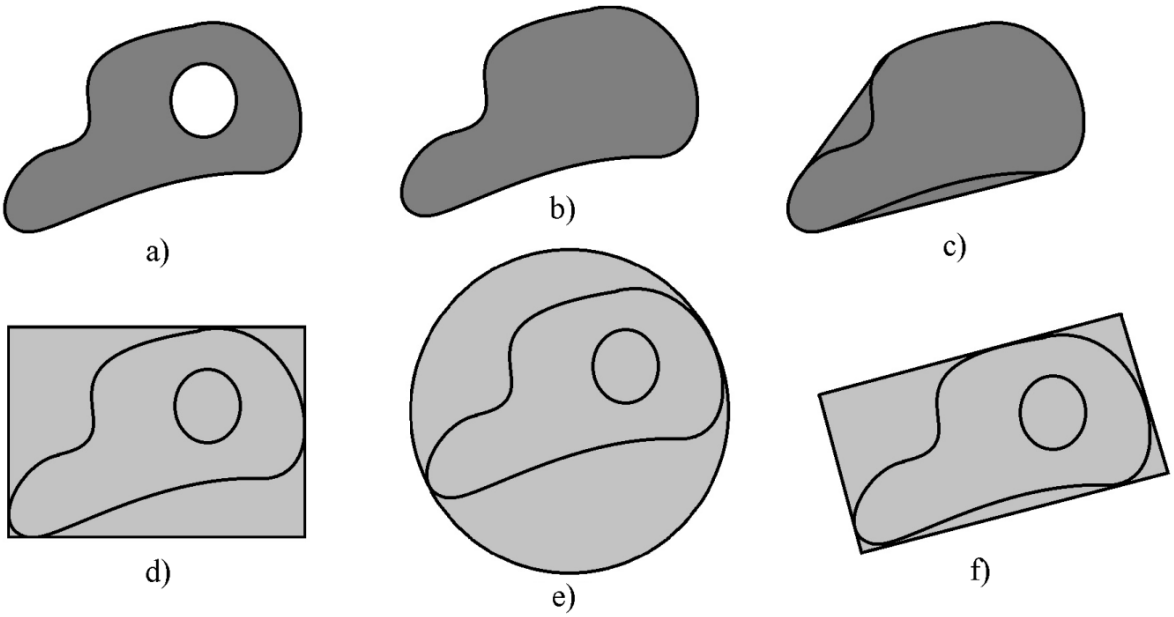


Figura 2.16: Diferencias entre: a) área Neta; b) área rellena; c) área convexa; d) área del minimo cuadrado que lo contiene e) área del circulo que lo contiene f) área del mínimo rectángulo que lo contiene.

realizar en objetos.

- Longitud de fibra: Es la longitud de la imagen total, obtenida de la esqueletonización.
- Ancho de fibra: Es la medida promedio de la distancia del esqueleto de la imagen hasta el borde.
- Largo: Máxima longitud de la figura convexa que cubre toda la imagen.
- Ancho: Si la imagen es irregular no es simple definir el ancho de un objeto, pero se puede definir como la mayor distancia perpendicular al largo.
- Perímetro: Es la longitud del contorno de un objeto o imagen.

- Medición de ubicación y orientación: Se puede obtener la locación de un objeto calculando la ubicación de su centro de masa o el punto medio del objeto. Un valor muy cercano a la locación es la orientación. Hay varias formas de definir la orientación ya sea midiendo el ángulo del eje mayor (la línea que une distancia

los dos puntos más alejados del objeto), el ángulo de su diagonal mayor (el eje mayor de la elipse que contiene al objeto), etc.

2.2.5. Descriptores

La idea principal de los descriptores es de alguna manera describir un objeto. En nuestro idioma se cuenta con pocos adjetivos para describir la forma de un objeto (largo, corto, arrugado, liso). Comúnmente se utiliza un parámetro de comparación para describir la forma ("se parece a... ", "tiene forma de..."). Encontrar descriptores numéricos de forma es difícil dado que no tienen una correspondencia con lo utilizado en la vida cotidiana. Para describir la forma de un objeto se utilizan valores que lo describan, los cuales no dependan ni de la locación (invariantes a la traslación)[84], de la orientación (invariantes a la rotación) [85, 86] ni del tamaño (invariantes a escala) . La forma básica de describir objetos se realiza con una simple combinación de parámetros (mediciones realizadas a la imagen) operados matemáticamente de manera que las dimensiones se cancelen. Por ejemplo la relación de aspecto $\frac{largo}{ancho}$ siempre permanece igual al cambiar el tamaño, la posición o la orientación del objeto. Dado que hay decenas de mediciones que se pueden realizar en una imagen, hay cientos de formas de combinar estos valores para obtener descriptores adimensionales. La Tabla 2.1 muestra algunos valores representativos de descriptores.

Otro tipo de descriptores de imágenes son los momentos de distintos órdenes de la imagen o de una región, los cuales se calculan sumando los pixeles y dándole un peso a cada uno (como la varianza, el sesgo o la curtosis). Estos parámetros se pueden calcular para ser invariantes ante la traslación, rotación o escala y ser utilizados para describir objetos.

2.2.6. Flujo óptico

El flujo óptico es una técnica utilizada para describir el movimiento aparente de objetos en una imagen causada por el movimiento del objeto o por el movimiento de la cámara. Se dice movimiento aparente, porque no siempre coincide con el movimiento real del objeto. La dirección del movimiento se puede conocer sabiendo

Factor de forma	$\frac{4 * \pi * Area}{Perimetro^2}$
Redondez	$\frac{4 * rea}{pi * (Ejemayor^2)}$
Relación de aspecto	$\frac{Ejemayor}{Ejemenor}$
Elongación	$\frac{Longituddefibra}{Anchodefibra}$
Curvatura	$\frac{Longitud}{Longituddefibra}$
Convexidad	$\frac{PermetroConvexo}{Permetro}$
Solides	$\frac{Area}{AreaConvexa}$
Compactación	$\frac{\sqrt{\frac{4}{\pi} * Area}}{EjeMayor}$
Extensión	$\frac{AreaNeta}{AreaRectanguloquelocontiene}$

Tabla 2.1: Tabla de descriptores.

donde se movió cada pixel de la imagen actual en la imagen siguiente. Dada una imagen I(x, y, t), donde la intensidad de cada punto depende del tiempo, para ver como cambia la imagen se deriva respecto al tiempo.

$$\frac{dI}{dt} = \frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} \tag{2.10}$$

Si se supone una iluminación constante, cada pixel mantendrá su luminosidad mientras se esté moviendo.

$$\frac{\partial I}{\partial t} = 0 \tag{2.11}$$

Si se reemplazan $\frac{dy}{dt}$ y $\frac{dx}{dt}$ por sus movimiento u y v.

$$\frac{dI}{dt} = \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v \tag{2.12}$$

Esta ecuación es la base de las técnicas de cálculo de flujo óptico, que identifica una línea perpendicular al gradiente de intensidad. Hay varios algoritmos para calcular el flujo óptico, los cuales calculan la función pixel a pixel, o con mayores restricciones como minimizar la diferencia en el gradiente de la imagen, o su aplicación a grupos de pixeles. El algoritmo más simple para calcular el flujo óptico es la comparación de los pixeles de la imagen t con los de la imagen (t+1). Si el pixel mantiene su luminosidad, el movimiento del pixel será el pixel que tenga la misma intensidad. La velocidad con a que un objeto se mueve ($\frac{dx}{dt}$), medida en ($\frac{pixel}{segundo}$) depende del rango de comparación. Si se compara solo con los pixeles vecinos, la velocidad máxima de movimiento será de 1 pixel por imagen. Si se compara a un radio de 2 vecinos, la velocidad máxima será de 2 pixeles por imagen capturada, y así sucesivamente. Como la intensidad lumínica no es siempre la misma, se calcula cuál de todos los vecinos tiene mayor probabilidad de ser el pixel original, siendo el más parecido el más probable. Este algoritmo también se puede aplicar a regiones, calculando los movimientos de grupos de pixeles. Cuanto más grande es la región de comparación mayor será la velocidad que se puede medir.

Las aplicaciones del flujo optico son muy variadas. Se puede utilizar en la segmentación de imágenes, reconociendo objetos diferentes debido a sus distintas velocidades, es posible realizar el seguimiento de objetos en movimiento, conocer la

velocidad de los objetos o de la cámara.

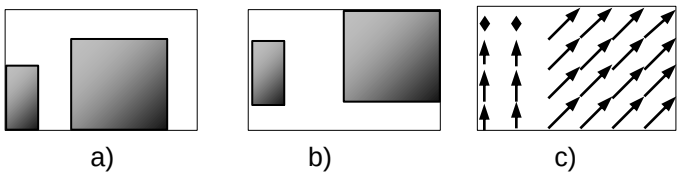


Figura 2.17: a)Imagen original b)Imagen siguiente b)Diagrama de flujo óptico.

2.2.7. Codificación en código cadena

Aunque la codificación en código cadena puede tomarse también como un descriptor, vale la pena explicarlo más en detalle. El código cadena es una forma más eficiente de representar una figura más allá de la enumeración de sus puntos. Esta forma de codificar la información radica en comenzar en un punto del dibujo e indicar en un arreglo las direcciones que se deben tomar para recorrer toda la imagen. Esta codificación seria como describir el movimiento de un lápiz necesario para realizar el dibujo. Estos movimientos se pueden describir de dos formas diferentes, de manera absoluta o relativa a la dirección del movimiento tal como se muestra en la Fig. 2.18. La codificación código cadena absoluta toma las direcciones referidas a un sistema de coordenadas fijo, siendo los valores que toma la codificación dependientes solo de la dirección del movimiento que se está realizando. La codificación relativa, indica los movimientos referidos a la dirección actual de movimiento, siendo los valores que toma la codificación dependiente del movimiento actual y del anterior. En la Fig. 2.19 se muestra una imagen y su representación en código cadena en formato absoluto y relativo.

2.2.8. Reconocimiento de objetos

El reconocimiento de objetos es la tarea de reconocer o encontrar un determinado objeto en una imagen. Esta tarea el hombre la realiza de manera muy rápida y robusta, pero aún no es realizada satisfactoriamente por los sistemas de visión

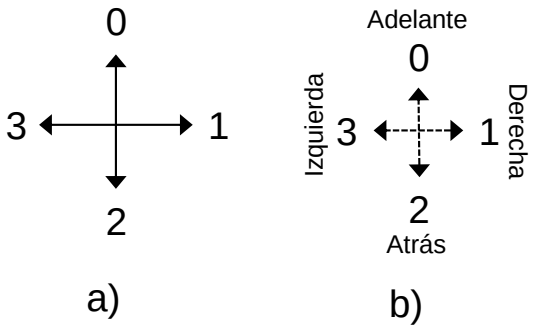
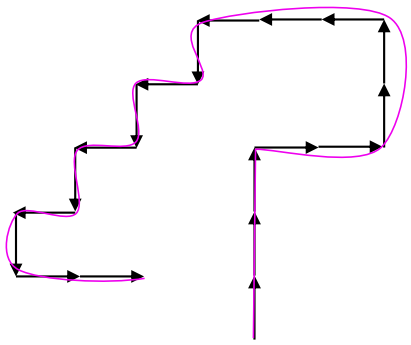


Figura 2.18: Valores de movimiento absoluto y relativo.

actuales. Los métodos actuales que resuelven este problema funcionan para un conjunto específico de objetos, tal como caras, caracteres, huellas digitales o vehículos en situaciones específicas, con un ángulo, iluminación y fondo determinado. El reconocimiento se puede separar en identificación y detección, donde la identificación es el proceso donde el objeto se encuentra ubicado y debe ser reconocido, por ejemplo identificación de una persona, etc. La detección es el proceso de buscar un determinado objeto o característica en la imagen por ejemplo buscar en una imagen de seguridad por armas o caras de personas particulares. La tarea de reconocimiento se realiza con toda la información obtenida con el procesamiento y análisis previo. Se utiliza la información obtenida con los algoritmos anteriores, ya sean las mediciones de los objetos, los descriptores, la información dada por la codificación código cadena, y con la utilización de herramientas matemáticas se realiza su procesado. Con toda la información obtenida se realiza la comparación, o búsqueda de similitudes, con una base de datos con información de lo que se quiere reconocer. Para realizar esta tarea se pueden utilizar varias herramientas matemáticas que se describen a continuación.

Minería de datos

La minería de datos (DM, por sus siglas en inglés, Data Mining) es el proceso de extraer información, o patrones, de un conjunto de datos. Es una técnica usada para todo tipo de análisis de información. La minería de datos se basa en algoritmos



a)00011003332323211

b) 0001030300313131330

Figura 2.19: Codificación en código cadena de una imagen a)Codificación Absloluta b)Codificación Relativa.

de muchas áreas, como de estadística, inteligencia artificial, redes neuronales, etc. Comúnmente, las tareas de minería de datos constan de los siguientes pasos:

- Agrupar: Descubrir grupos o estructuras que de alguna manera son similares o comparten alguna característica en común.
- Clasificar: Generalizar la estructura de datos, para colocar la nueva información en los diferentes grupos.
- Regresión: Encontrar una función que modele la información con un mínimo error.
- Asociación: Buscar relaciones entre las variables.

Entre los métodos que se utilizan para realizar la minería de datos se pueden nombrar: Análisis por componentes principales, Análisis por componentes independientes, Análisis de regresiones, Análisis bayesiano, Análisis combinatorio, Redes neuronales, Modelo oculto de markov, etc.

Redes neuronales

Las redes de neuronas artificiales (RNA), son un paradigma de aprendizaje y procesamiento automático inspirado en la forma en que funciona el sistema nervioso de los animales. Se trata de un sistema de interconexión de neuronas en una red que colabora para producir un estímulo de salida. Estos sistemas están compuestos por elementos básicos (Neuronas), agrupados en capas que se encuentran interconectados entre sí (Sinapsis). Esta estructura posee varias entradas y salidas, y varios niveles de neuronas. La Fig. 2.20 muestra una red neuronal de varias capas con las neuronas y las interconexiones. Como un ejemplo de red neuronal se puede mencionar el *Mapa auto organizado (Self organized Map)*, también llamado red de Kohonen. Este es un tipo de red neuronal no supervisado, competitivo, distribuido en forma regular en una rejilla, normalmente de dos dimensiones, cuyo fin es descubrir la estructura subyacente de los datos introducidos en ella. Otro ejemplo de red neuronal es la llamada Red Celular no lineal, que se explicará en detalle en el capítulo 3

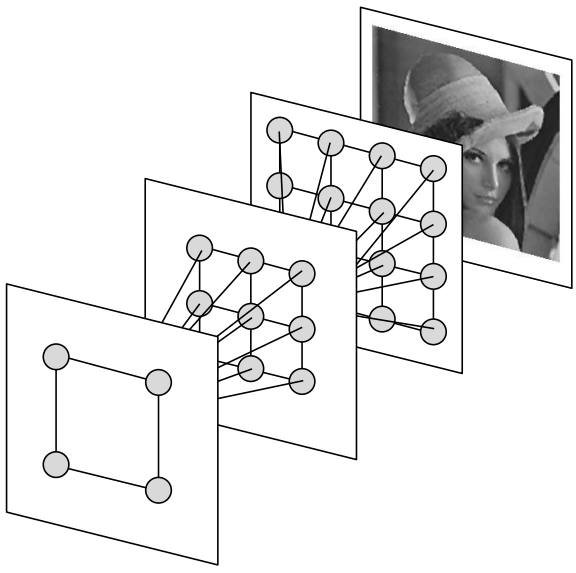


Figura 2.20: Red neuronal de varias capas.

Análisis con transformadas

El análisis de imágenes con transformadas permite trabajar con las imágenes en un dominio diferente del espacial. Las transformadas se pueden utilizar para filtrar imágenes, reconocer objetos, codificar la imagen, etc. Algunas de las transformadas más comunes utilizadas en el procesamiento de imágenes son: transformada de Fourier, transformada del coseno, transformada del seno, transformada de Karhunen-Loeve, Hadamard, transformada de Hugh, transformada de Abel

Análisis fractal

Un fractal es un objeto formado por una estructura básica que se repite a diferentes escalas y rotaciones. Esta técnica se puede utilizar para comprimir imágenes, modelar formas naturales o analizar imágenes teniendo en cuenta su dimensión fractal. Se pueden comprimir imágenes para disminuir la cantidad de información a procesar, encontrar patrones, o autosimilitudes y reconstruirla separando la imagen en pequeñas unidades básicas de información. Esta es un área joven y sin muchas herramientas aún.

2.3. Resultados existentes

Desde la aparición de las cámaras en tecnología CMOS y CCD, se han desarrollado una gran cantidad de sistemas integrados de diversa naturaleza de funcionamiento [66, 67, 68, 73]. Un análisis de la literatura y de los productos disponibles en el mercado de consumo revela varios enfoques posibles. Si bien resultan difícil de clasificar, se podrían separar en base a ciertas características sobresalientes como por ejemplo: sistemas Blanco y Negro o en escala de grises/color; sistemas de alta velocidad o de bajo consumo; sistemas de procesamiento espacial o espacio-temporal; sistemas en procesos CCD, CMOS estándar o CMOS 3D; sistemas de procesamiento digital o analógico, etc. Todas estas categorías de integrados poseen características particulares y ninguno de los integrados que se encuentran desarrollados puede encasillarse dentro de una sola de las categoría.

Hay varios enfoques para implementar pre-procesamiento de imágenes en el mismo chip que capta las imágenes. En principio se puede contar con una estructura de procesamiento externa (no perteneciente al arreglo de sensores), que tome la información y la analice [80]. También se pueden colocar varios procesadores, que operen a nivel de fila (o columna) o a nivel de arreglo [81]. Otro caso es la realización de píxeles inteligentes, donde los procesadores se encuentran integrados en el mismo píxel. Este enfoque permite generar un procesamiento paralelo por píxel real, a expensas de un mayor tamaño de píxel. Sin embargo, los avances en tecnología CMOS, permiten realizar dispositivos cada vez más pequeños. Dentro de este enfoque pueden agruparse los circuitos que están basados en una estructura del tipo red no-lineal celular o red neuronal celular, conocida en la literatura como CNN ver 3. Esta estructura fue originalmente introducida por Chua en [9, 8] y se basa en una celda que evoluciona de acuerdo a la ley de un sistema dinámico no-lineal con conexiones a celdas vecinas. Si bien todas las celdas del arreglo son idénticas, cambiando los coeficientes del sistema no lineal se pueden lograr diversas tareas sobre una imagen de entrada . Otro enfoque es el de los procesadores de instrucción simple (SIMD). Los mismos también tienen arreglos de celdas y en cada celda bloques de operaciones lógicas y de transferencia de datos. Un programa almacenado en memoria contiene las instrucciones que cada bloque de la celda debe ejecutar. Los procesadores del tipo SIMD disponibles poseen características bastante disímiles, las cuales son resultado de la aplicación particular para las cuales fueron diseñados Otro enfoque posible es el de sistemas integrados inspirados en la naturaleza [66, 67, 71, 77], llamados retinas electrónicas[75, 76] . A continuación se describen algunos circuitos integrados con capacidades de procesamiento.

MIPA4K

En los trabajos [35, 44] se presenta un procesador de señal mixta llamado MIPA4K. Este circuito integrado es un arreglo de 64×64 que combina operaciones de escala de grises con procesamiento de imágenes binarias. Cada una de las celdas del arreglo contiene un fotodiodo, un conversor A/D dos conversores D/A varios tipos

de filtros (de rango y anisotropico) bloques de memoria digital y analógica, y hardware para procesar imágenes binarias. Este circuito está equipado con la opción de generar una ventana de interés o ROI (por sus siglas en inglés, Region of interest).

ASPA y SCAMP

En los trabajos [40, 20, 22] el grupo de trabajo del laboratorio de diseño microelectrónico de la universidad de Manchester, formado por Piotr Dudek, Stephen Carey y colaboradores, han diseñado varios chips de arreglo de procesadores y de visión [24, 21, 39]. Entre los que se encuentra el ASPA (Asynchronous/Syncronous Cellular Processor Array) y el SCAMP (SIMD Current mode Analog Matrix Processor).

El ASPA es un chip de visión digital que opera de modo sincrónico y asincrónico formado por un arreglo ortogonal de procesadores llamados PE (por sus siglas en inglés, Processing Element) los cuales ejecutan la misma instrucción coordinada por un controlador central. Cada una de las celdas posee un microprocesador que consiste de una ALU, ocho registros de ocho bits, (64 bits de memoria por celda), un foto transductor, un convertir A/D, registros auxiliares y una unidad de comunicaciones encargada de multiplexar la información de entrada. Cada uno de los PE se comunica con sus cuatro vecinos en forma de cruz (+) enviando y recibiendo información de manera local. Además cada celda posee un bloque de "propagación"que la propagación de la onda asincrónica a través de todo el arreglo. La topología de la red de propagación es controlada por información local en cada pixel. De esta manera se pueden producir ondas disparadoras que se propagan a través de la red, y que realizan operaciones globales (como rellenado de agujeros) de manera muy rápida.

El SCAMP es un circuito de procesamiento paralelo del tipo SIMD programable por software formado por una red de 21×21 procesadores (PE) que ejecutan instrucciones enviadas por un controlador digital. Cada uno de los PE posee un foto transductor, una unidad aritmética lógica, un banco de registros y puertos de comunicación globales y locales (con los PE vecinos). El procesador puede realizar operaciones de suma, resta, multiplicación y división. El procesamiento es llevado a cabo a raves de señales analógicas muestreadas usando circuitos con técnicas de

llaveado de corriente (switch current). A la salida del integrado se realiza la lectura por fila o por columna en paralelo y se dispone de conversores analógicos digitales dando una salida digital de ocho bits.

ACE16K y CNN Multicapas

En los trabajos [45, 2, 38, 37] y [18]presentados por Rodriguez-Vazquez, Dominguez-Castro y colaboradores, de la Universidad de Sevilla y al Instituto de Microelectrónica de Sevilla se introducen dos integrados uno con procesamiento en el plano focal denominado "ACE16K" y un prototipo con tres capas de procesamiento.

El "$ACE16K$" es un arreglo de 128×128 celdas con un procesador CNN estándar (ver Sec. (3.2)) analógico. Cada una de las celdas del arreglo contiene un convolucionador CNN, un procesador lógico elemental, un banco de almacenamiento y un detector de umbral para el direccionamiento por eventos. El integrado puede trabajar en tres modos de integración lineal de corriente fotónica -dando una salida en corriente- o en tres modos alternativos de salida en tensión -realizando una compresión logarítmica de la imagen. Un rasgo sobresaliente del ACE16k es que todos los pesos y entradas necesarias para la selección de un programa CNN junto con el ingreso de imágenes se realiza en forma de palabras digitales ya que posee un banco interno de conversores digital a analógico para todas sus líneas de control. Esta facilidad permite el almacenamiento de varios programas CNN en su interior en un banco de memoria digital facilitando la utilización de la cámara inteligente en tareas de mayor complejidad. Las salidas del integrado son digitalizadas en el bloque de I/O por medio de un banco de conversores A/D dispuestos por columna en la lectura del arreglo. Los resultados de las investigaciones realizadas en este grupo llevaron a la creación y comercialización de varios circuitos integrados posteriores al ACE16K a través de la empresa ANAFOCUS (www.anafocus.com).

El prototipo fabricado pose un arreglo de 32×32 con un procesador del tipo CNN Universal Machine. Cada celda posee tres niveles de procesamiento, con una memoria digita, una memoria analógica y una unidad lógica de procesamiento local. Los parámetros que determinan el comportamiento de la red se distribuyen a todas las celdas a través de buses globales. Cada una de las celdas posee un tiempo de

integración programable configurable con parámetros globales y locales para lograr una captura de imagen adaptiva.

Xenon

En el trabajo [28] de Péter Földesy, Ákos Zarándy, Csaba Rekeczky y sus colaboradores en el instituto de investigación en computación y automatización en la "Hungarian Academy of Sciences"muestran el Xenon v3. Este circuito integrado es un arreglo de 8×8 celdas con un procesamiento del tipo SIMD. Cada una de estas celdas contiene un arreglo de 8×8 fotosensores, un procesador aritmético, un procesador morfológico, un banco de registros y dos bloques de comunicaciones (interna y externa). La unidad aritmética contiene un procesador, con un acumulador de 24 bits. Esta unidad permite calcular suma, resta, multiplicación y comparación de valores de 8 o 16bits. Las imágenes en blanco y negro se operan con el procesador morfológico, el cual contiene 8 procesadores morfológicos de 1 bit, que calcula operaciones morfológicas. Cada celda posee 512bytes de memoria para almacenar datos de 16bits, 8 bits o 1 bit. Todas las celdas ejecutan la misma instrucción, definida por un controlador externo que envía parámetros y atributos a las celdas, pudiéndose enmascarar algunas celdas para habilitar o deshabilitar la ejecución de ciertos comandos.

iVisual

En el trabajo [4, 3]presentado por Chih-Chi Cheng en [4] y colaboradores detallan un sistema de adquisición y procesamiento de imágenes del tipo SIMD, con un arreglo de 128 procesadores. Este sistema está diseñado para trabajar con imágenes de video. Los procesadores no forman un arreglo junto con los sensores, sino que están distribuidos en el chip. El sistema se compone de un arreglo de fotosensores, una memoria, un procesador global, una unidad de procesamiento de características y una unidad de decisiones. En la memoria se guardan los valores de lectura de los sensores CMOS, y todas las unidades de procesamiento tienen acceso a ella. El procesador global está formado por 128 procesadores que trabajan en paralelo. El Procesador de características procesa la información que ingresa en el de forma

paralela y tiene solo una salida escalar (Muchas entradas una salida).La unidad de decisiones es un procesador del tipo MIPS (que corre millones de instrucciones por segundo) de 32 bits con un pipeline de 5 etapas. Su entrada puede ser la memoria, la entrada o la salida del bloque procesador de características. La unidad de decisiones también controla la ejecución de programa de los otros dos procesadores.

S-CNN

En el trabajo [14] presentado por Mandolesi y colaboradores se muestra un circuito integrado que realiza un procesamiento digital de imágenes del tipo S-CNN (Ver 3) que es un tipo particular de red celular no lineal donde la función que ejecuta cada celda está definida por una función lineal a tramos. Este circuito integrado está formado por un arreglo hexagonal de 7×6 pixeles, donde cada uno posee una vinculación con sus siete vecinos. Cada celda posee un fotoconversor, un conversor A/D, un banco de registros de 7 bits distribuido, y una unidad de procesamiento central. El fotoconversor activo (APS) está formado por un fotodiodo con una isla de implante n sobre el sustrato de tipo p. El conversor A/D es del tipo rampa simple, para el cual se utiliza un comparador del tipo conmutado. Se realiza una codificación en tiempo de las señales para realizar el procesamiento de la información, siendo el tiempo de procesamiento proporcional a la precisión de los registros. Este procesador puede realizar operaciones binarias, morfológicas, y puede realizar filtrado no lineal de la imagen.

SRVC

En el trabajo presentado por Wei Miao [42], se presenta un circuito integrado con una estructura de procesamiento SIMD. Este chip está formado por un arreglo de 16×16 procesadores, combinado con procesadores de fila y de columna. Cada una de las celdas del arreglo está formada por un fotodiodo, un conversor A/D de 1 bit, un procesador de 1bit, una ALU y una memoria de 4bits. Cada celda esta comunicada con sus cuatro vecinas en forma de cruz (). Este circuito es un procesador de imágenes de 1 bit. Puede realizar operaciones lógicas y morfológicas en la imagen. Puede calcular el rango de una imagen, obteniendo una región de interés (ROI).

2.4. Conclusión

En este capítulo se han descripto algunos de los algoritmos necesarios para procesar, analizar y extraer información de imágenes. Como tareas de bajo nivel se ha mostrado como realizar el mejoramiento de imágenes, se han descripto tareas básicas de procesamiento de imágenes binarias. Como tareas de medio nivel se han mostrado algunos métodos de segmentación, se han nombrado algunas de las mediciones que se pueden realizar y como obtener de ella descriptores. Como tareas de alto nivel se ha explicado lo que es el flujo óptico y como calcularlo, la codificación código cadena, y como por medio de herramientas estadísticas se pueden utilizar los descriptores para realizar reconocimiento de objetos. En la segunda parte de este capítulo se han mostrado algunos circuitos integrados que realizan procesamiento de imágenes que se encuentran actualmente en desarrollo.

Capítulo 3

Redes Celulares No Lineales

La Red Celular No Lineal (o CNN por sus siglas en inglés - Cellular Neural Network) introducida originalmente por CHUA [11, 48] es una arquitectura compuesta de celdas interconectadas que permite realizar procesamiento en paralelo de la información. En la CNN, cada célula o celda implementa un sistema dinámico no-lineal y el procesamiento se realiza por la acción coordinada de todas ellas (/cite CHUAvision). De esta manera, a partir de la acción de celdas individuales con conectividad resulta posible implementar tareas de cálculo complejas. Las CNN bidimensionales constituyen una atractiva arquitectura sobre la que se pueden implementar varios procesos para extraer información [12] o realizar una interpretación de la imagen. Esta arquitectura se puede desarrollar en formato analógico [32] o digital, siendo la segunda opción atractiva por sus propiedades de escalabilidad. Las CNN son una herramienta poderosa para visión y procesamiento de imágenes y de video[17] . Este capítulo tiene como objetivo dar una descripción de las Redes celulares no lineales y presentar el caso especial de la CNN Simplicial [31], describiendo el algoritmo que permite calcular la evolución de la celda.

A continuación, se describe brevemente la CNN lineal o CNN estándar, que ha sido ampliamente utilizada para realizar procesamiento de imágenes ([47], [7].), entre las que se destacan la máquina universal CNN [13] y las cámaras CMOS con estructura de procesamiento CNN [16].

3.1. Introducción

La red CNN fue originalmente introducida como un arreglo espacial de celdas con vinculaciones locales. Cada una de estas celdas es un sistema dinámico que evoluciona según una función que depende de los estados y entradas de la celda y sus vecinos. Cada una de estas celdas incluye también una función de salida que tiene como argumento de entrada su estado. Las celdas directamente vinculadas en la red se influencian en forma directa; las celdas más alejadas no poseen conexión directa pero son influenciadas con el correr del tiempo cuando la información se propaga por las celdas intermedias. En general las variables asignadas a cada celda son tres: *entrada*, *estado* y *salida*, tal como lo muestra la Fig. 3.1.

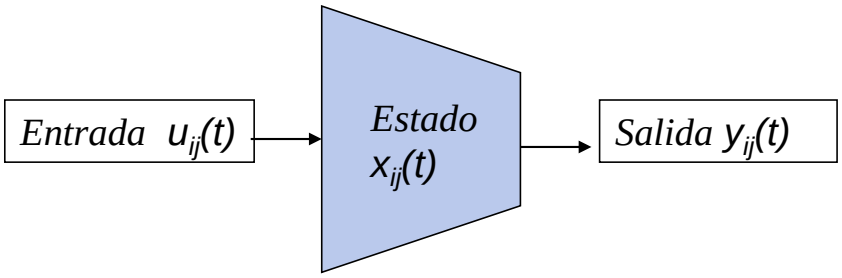


Figura 3.1: Diagrama de una celda aislada

La descripción matemática de una celda está dada por la ecuación que relaciona la dinámica del estado con las entradas y salidas, y por la ecuación que especifica el mapeo del estado interno en la salida:

$$\begin{cases} \dot{x}(t) = F(\mathbf{y}_S(t), \mathbf{u}_S(t)) \,, \\ y(t) = G(x(t)) \end{cases} \tag{3.1}$$

donde la función $F : \Re^m \to \Re^1$ determina la dinámica de la celda y $G : \Re^m \to [0, 1]$ es la función de salida. Los vectores $\mathbf{y}_S(t) \in \Re^n$ y $\mathbf{u}_S(t) \in \Re^n$ son los vectores formados por las salidas y entradas de las celdas vecinas y de la propia celda.

Las celdas se identifican en la estructura o red por medio de subíndices que las localizan en el espacio en forma unívoca. En el caso de una red de dos dimensiones,

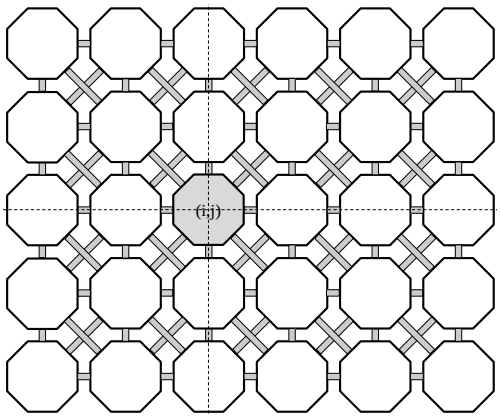


Figura 3.2: Identificación de una celda en el arreglo

la celda con subíndices i y j, se denomina celda $C_{i,j}$, como se muestra en la Fig. 3.2. Si bien la conectividad y la ecuación dinámica de cada celda podrían ser diferentes a lo largo del arreglo, en general, y sobre todo en el procesamiento de imágenes, la estructura se hace homogénea; es decir, todas las celdas tienen la misma conectividad e implementan la misma función. Si se identifica una de estas celdas como $C_{i,j}$, (3.1) se puede escribir como:

$$\begin{cases} \dot{x}_{i,j}(t) = F(\mathbf{y}_S(t), \mathbf{u}_S(t)) \,, \\ \\ y_{i,j}(t) = G(x_{i,j}(t)) \end{cases} \tag{3.2}$$

Al conjunto de celdas involucradas en la evolución del estado se la denomina *esfera de influencia*, y se lo nota $S_{i,j}$. La descripción de la esfera de influencia se puede realizar enumerando el conjunto de elementos que la componen o en base a una propiedad basada en la distancia al elemento central:

$$S_{i,j} = \{C_{i-1,j-1}, C_{i-1,j}, \ldots, C_{i,j}, \cdots, C_{i+1,j+1}\} \,. \tag{3.3}$$

En la Fig. 3.3 se muestran dos esferas de influencias determinadas por el conjunto de celdas dentro de un radio de separación $r = 1$ y $r = 2$ (Fig. 3.3a y Fig. 3.3b, respectivamente), donde la distancia se mide con respecto a la celda central $C_{i,j}$ utilizando la norma infinito.

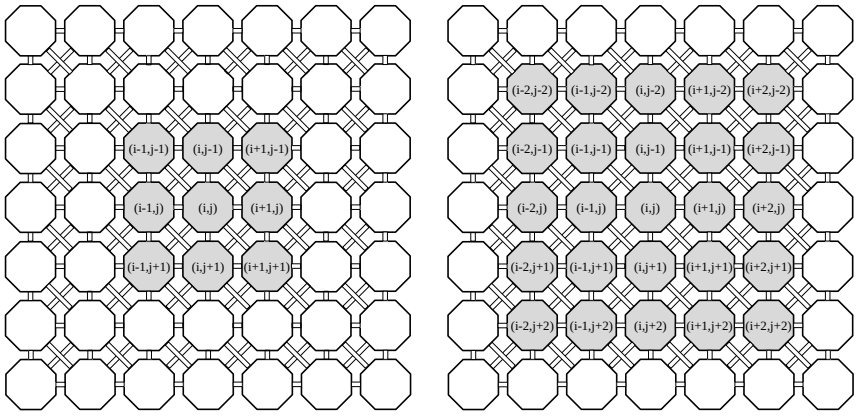


Figura 3.3: a) Esferas de influencia de radio 1; b) Esferas de influencia de radio 2

De acuerdo a la Ec. (3.2) todas las entradas $u_{k,l}$ y todas las salidas $y_{k,l}$ de las celdas $C_{k,l}$ de la esfera de influencia componen los vectores de entrada $(\mathbf{y}_S, \mathbf{u}_S)$ a la celda $C_{i,j}$.

La ecuación de evolución no está completamente definida en los bordes del arreglo, ya que las celdas en los extremos no tienen vecinos, y la esfera de influencia se extiende por fuera del arreglo. Se definen entonces condiciones de borde para que la ecuación esté definida, y las celdas tengan una esfera de influencia completa. La Fig. 3.4 muestra las tres condiciones de bordes más usadas.

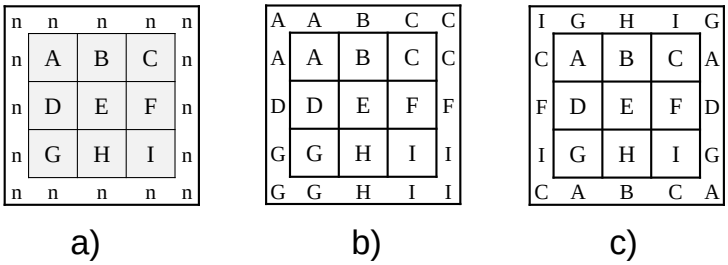


Figura 3.4: condiciones de borde de arreglos, a)Condicion Fija, b) Neumann c) Toroidal

- Fija (Dirichlet): A cada uno de las celdas que se encuentra fuera del arreglo,

se le asigna un valor fijo.

- Flujo cero (Neumann) : A cada una de las celdas fuera del arreglo, se le asigna un valor igual a la de su vecino dentro del arreglo.
- Periódica (Toroidal): Cada uno de los vecinos fuera del arreglo, es idéntico al del borde contrario.

Habiendo definido la expresión de una celda aislada y de su esfera de influencia, es claro que la función particular $F(\cdot)$ que se utilice definirá las características del sistema. Esto a su vez, definirá la tarea a realizar por la CNN.

3.2. Expresión de la CNN estándar

La estructura de la CNN estándar [9] fue la primer CNN presentada y también es la más utilizada en toda la literatura especializada. La ecuación de estados que rige la dinámica de una celda aislada de la CNN estándar es:

$$\dot{x}_{i,j} = -x_{i,j} + a_{i,j}y_{i,j} + b_{i,j}u_{i,j} + z_{i,j} \tag{3.4}$$

donde $a_{i,j} \in \Re^1$ y $b_{i,j} \in \Re^1$ son coeficientes de peso y

$$y_{i,j} = f(x_{i,j}) = \frac{1}{2}(|x_{i,j}+1| - |x_{i,j}-1|) = \begin{cases} 1, & x_{i,j} \geq 1 \\ x_{i,j}, & |x_{i,j}| < 1 \\ -1 & x_{i,j} \leq 1 \end{cases}$$

es una función del tipo saturación que se aplica sobre el estado de la celda, y $z_{i,j} \in \Re^1$ es un valor de referencia o "$bias$".

Si se amplía la expresión de la celda completamente aislada, incluyendo los vínculos locales con otros elementos de la red, (3.4) se convierte en:

$$\dot{x}_{i,j} = -x_{i,j} + A(\mathbf{y}_S) + B(\mathbf{u}_S) + z_{i,j}, \tag{3.5}$$

donde los operadores $A : \Re^n \to \Re^1$ y $B : \Re^n \to \Re^1$ representan las sinapsis de "feedback" y "feedforward" que relacionan las salidas y entradas de las celdas de la esfera de influencia con el estado de la celda $C_{i,j}$. En el caso de la CNN estándar,

$$B(u_{i,j}) = \sum_{k,l \in S_{i,j}} b_{k,l} u_{k,l}$$

$$A(y_{i,j}) = \sum_{k,l \in S_{i,j}} a_{k,l} y_{k,l},$$

con lo cual, la ecuación de estados resulta de la siguiente forma:

$$\dot{x}_{i,j} = -x_{i,j} + z_{i,j} + \sum_{k,l \in S_{i,j}} a_{k,l} y_{k,l} + \sum_{k,l \in S_{i,j}} b_{k,l} u_{k,l}. \tag{3.6}$$

La ecuación de salida mantiene su expresión:

$$y_{i,j} = \frac{1}{2}(|x_{i,j} + 1| - |x_{i,j} - 1|).$$

Representación por Patrones y por Genes CNN

Si se reescribe la expresión de la CNN estándar, dada por (3.6) en forma matricial, se obtiene la siguiente expresión:

$$\dot{x}_{i,j} = -x_{i,j} + z_{i,j} + \mathbf{A}\mathbf{y}_{S_{i,j}} + \mathbf{B}\mathbf{u}_{S_{i,j}}, \tag{3.7}$$

donde las matrices $\mathbf{A}$ y $\mathbf{B}$ corresponden al mapeo de las salidas y entradas sobre el estado y se denominan comúnmente -junto con el valor de referencia z- patrones o "templates" de la CNN. Por ejemplo, si se considera una CNN estándar con una esfera de influencia de radio $r = 1$,

$$S_{i,j}(1) = \{C_{i-1,j-1}, C_{i-1,j}, \ldots, C_{i,j}, \cdots, C_{i+1,j+1}\}$$

el "template" o patrón que define a la CNN estándar está dado por las matrices

$$\mathbf{A} = \begin{pmatrix} a_9 & a_8 & a_7 \\ a_6 & a_5 & a_4 \\ a_3 & a_2 & a_1 \end{pmatrix}$$

y

$$\mathbf{B} = \begin{pmatrix} b_9 & b_8 & b_7 \\ b_6 & b_5 & b_4 \\ b_3 & b_2 & b_1 \end{pmatrix}$$

y por el valor del "bias" $z_{i,j}$, donde $a_1, \ldots, a_9$ son los coeficientes asociados a las sinapsis de realimentación y $b_1, \ldots, b_9$ al de la entrada de las celdas $C_{i-1,j-1}$,$C_{i-1,j}$,$\ldots$, $C_{i,j}$,$\cdots$,$C_{i+1,j+1}$ respectivamente. Para reducir la notación de los patrones se reordenan los valores de las matrices y se compone un único vector llamado *gen*. En el ejemplo anterior, el gen que describe la CNN de radio $r = 1$ tiene la forma:

$$[z, b_9, b_8, b_7, b_6, b_5, b_4, b_3, b_2, b_1, a_9, a_8, a_7, a_6, a_5, a_4, a_3, a_2, a_1].$$

La CNN estándar ha sido ampliamente estudiada y se conoce una gran cantidad de genes que resuelven problemas especiales. Para el caso de una CNN estándar aplicada al procesamiento de imágenes, donde a celda se asocia con un pixel, se pueden encontrar bibliotecas de genes utilitarios que resuelven una inmensa cantidad de problemas de procesamiento lineal, no lineal, identificación de patrones, etc. En [6] y [10] se listan bibliotecas de genes útiles en el procesamiento de imágenes. Esta descripción fue además adoptada en la mayoría de las realizaciones integradas, por ejemplo la CNN-UM [13], [16], [46], etc.

3.3. Expresión de la CNN Simplicial o S-CNN

La S-CNN es un caso particular de CNN presentada por Julián en [31] donde la dinámica que rige el cambio de estado de las celdas, está definida por una relación lineal a tramos o "PWL" (por sus siglas en inglés, Piecewise-Linear). La ecuación de la S-CNN es:

$$\begin{cases} \dot{x}_{i,j}(t) = F(\mathbf{y}_{S_{i,j}}(t), \mathbf{u}_{S_{i,j}}(t)), \\ y_{i,j}(t) = x_{i,j}(t), \quad i = 1, \cdots, N \quad j = 1, \cdots, M; \end{cases} \tag{3.8}$$

donde $F : \Re^m \rightarrow \Re^1$ es una función lineal a tramos; $\mathbf{y}_{S_{i,j}}(t) \in [0,1]^n$ es el vector de salidas de las celdas pertenecientes a la esfera de influencia $S_{i,j}$; $\mathbf{u}_{S_{i,j}}(t) \in [0,1]^n$ es el vector de entradas en $S_{i,j}$; $x_{i,j}(t) \in [0,1]$ es el estado de la celda $C_{i,j}$; N y M son el total de filas y columnas de la S-CNN n es el número de celdas que integran la esfera de influencia y $m = 2n$ es la dimensión del espacio conjunto de entradas y salidas.

La función F es una función PWL definida sobre una partición simplicial del espacio formado por el conjunto de entradas y salidas ($\Re^m$) de la CNN, como se puede ver en la Ec. (3.8). Un dominio simplicial, como su nombre lo indica, está dividido en símplices, siendo un símplice la generalización de un triángulo como figura geométrica en $\Re^2$. Más exactamente, un símplice es la envoltura convexa de un conjunto de (n + 1) puntos independientes afines en un espacio euclídeo de dimensión n o mayor. En cada símplice la función PWL es una función lineal afín que se puede representar con un hiperplano -válido solamente sobre este símplice. El hiperplano queda determinado por la combinación convexa de los $m + 1$ valores de función en cada vértice del símplice. Por ejemplo, la envoltura convexa de 2 puntos es una línea, la de tres puntos es un triángulo (que forma un plano), la de cuatro puntos, es una pirámide (que forma un hiperplano). Los vértices son los puntos dentro del dominio definidos por las intersecciones de dimensión cero de la partición. Si se selecciona una partición unitaria del dominio (un espacio m-dimensional con valores ente 0 y 1), el conjunto de vértices del mismo está dado por:

$$V_D = \{ \mathbf{v} \in \Re^m : v_i \in \{0,1\}; i = 1, \cdots, m\}. \tag{3.9}$$

y el conjunto de símplices correspondientes a esta partición es:

$$\begin{aligned} \widetilde{V}_D = \Big\{ & \widetilde{V}_D^k = \{\mathbf{v}^1, \cdots, \mathbf{v}^i, \cdots, \mathbf{v}^{m+1}\} : k = 1, \cdots, m!, \\ & v_j^1 = 0,\ j = 1, \cdots, m; \quad v_j^{m+1} = 1,\ j = 1, \cdots, m; \\ & \|\mathbf{v}^{i+1} - \mathbf{v}^i\|_\infty = 1,\ i = 1, \cdots, m \Big\}. \end{aligned} \tag{3.10}$$

Una representación alternativa que favorece la realización integrada de la S-CNN es la propuesta por Julián y otros en [30]; la función PWL se expresa en un espacio

de funciones donde la base está formada por funciones PWL con características especiales. Cada elemento $\alpha_i : \Re^m \to \Re^1$ de la base toma un valor numérico igual a uno sobre un vértice y cero en todos los restantes:

$$\alpha_i(\mathbf{v}_j) = \begin{cases} 1 & \text{si } i = j \\ & \mathbf{v}_j \in V_D \quad i,j = 1,\cdots,q\,. \\ 0 & \text{si } i \neq j \end{cases}$$

El conjunto de elementos α_i $i = 1,\ldots,q$ forman la base Λ,

$$\Lambda(\cdot) = [\alpha_1(\cdot),\cdots,\alpha_q(\cdot)]$$

de funciones "PWL" y en términos de esta base la función descriptora F de la S-CNN queda expresada como la combinación lineal:

$$F(w) = c^T \Lambda(w)\,,$$

donde $w \in \Re^m$ es un punto en el espacio conjunto de entrada y salida; $c \in \Re^q$ es el vector de coeficientes de la función F en la base Λ, donde $q = 2^m$ es el número de vértices de la partición simplicial. En esta expresión el vector de parámetros coincide con los valores de la función en cada vértice. Esta información puede almacenarse en una tabla de memoria donde a cada entrada de la memoria se le asocia un vértice y el campo almacenado es el valor de la función en el vértice. La gran ventaja de esta representación es que para un punto arbitrario perteneciente a un dado símplice, el cálculo de la función solo requiere de la evaluación de $q+1$ funciones de la base, y sus coeficientes asociados, que se extraen de la memoria. A su vez, cada función de la base tiene una expresión en extremo sencilla de calcular. Nótese que q solo depende de la dimensión del dominio, y no de la extensión de la partición.

A continuación se introduce la expresión discreta de la S-CNN que permite una realización puramente digital del procesador S-CNN:

$$\begin{cases} x_{i,j}(k+1) = F(\mathbf{y}_{S_{i,j}}(k), \mathbf{u}_{S_{i,j}}(k))\,, \\ y_{i,j}(k) = x_{i,j}(k), \quad i = 1,\cdots,N \quad j = 1,\cdots,M \end{cases} \tag{3.11}$$

donde $F : \Re^m \to \Re^1$ es una función PWL; $\mathbf{y}_{S_{i,j}}(k) \in [0,1]^n$ y $\mathbf{u}_{S_{i,j}}(k) \in [0,1]^n$ son los vectores de salidas y estados de las celda de la esfera de influencia $S_{i,j}$; $x_{i,j}(k) \in [0,1]$ es el estado de la celda $C_{(i,j)}$; N y M son los números de filas y columnas y n es el número de celdas que componen la esfera de influencia.

3.4. Cómputo de una red S-CNN

Como se vió en la sección previa, el cómputo de una red S-CNN requiere la evolución temporal del sistema dinámico discreto:

$$\begin{cases} x_{i,j}(k+1) = F(\mathbf{y}_{S_{i,j}}(k), \mathbf{u}_{S_{i,j}}(k)) \,, \\ \\ y_{i,j}(k) = G(x_{i,j}(k)) \,, \quad i = 1, \cdots, N \quad j = 1, \cdots, M \end{cases} \tag{3.12}$$

donde $F : \Re^m \to \Re^1$ es una función lineal a tramos; $G : \Re^1 \to [0,1]$ es la función de salida; $\mathbf{y}_{S_{i,j}}(k) \in \Re^n$ y $\mathbf{u}_{S_{i,j}}(k) \in \Re^n$ son los vectores de salidas y entradas, respectivamente, de las celdas pertenecientes a la vecindad $S_{i,j}$; $x_{i,j}(k) \in \Re$ es el estado de la celda (i,j); N y M son el número total de filas y columnas de la red y n es el número de celdas en la esfera de influencia.

Para calcular la evolución de la red es necesario evaluar la función de transición de F, simultáneamente en todas las celdas. Si nos restringimos a las S-CNN discretas que se detallaron en la sección anterior el cómputo de la función F se puede realizar por medio de la interpolación lineal de la función en los puntos extremos de un símplice. En este caso, la forma de calcular la función de transición consiste en encontrar dentro del dominio de la función el símplice que contiene el punto y realizar una interpolación lineal sobre el rango (ver Fig. 3.5). El procedimiento de cálculo se puede resumir mediante el siguiente algoritmo:

Paso 1 : Dado un vector de entrada w, se debe encontrar el símplice que lo contiene, lo que es equivalente a encontrar el conjunto de vértices que lo determinan (Fig. 3.5 b)).

Paso 2 : Una vez encontrado el símplice, se debe descomponer el vector de entrada como una combinación lineal convexa de los vértices del símplice; es decir, se

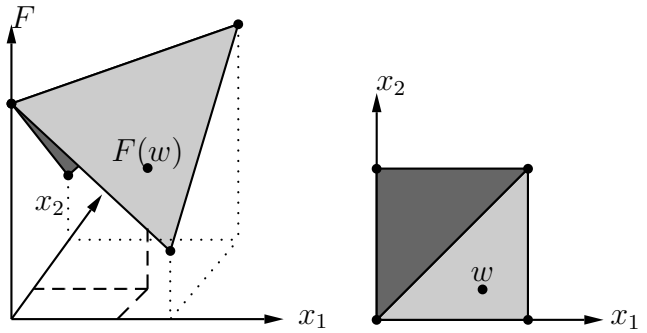


Figura 3.5: a) Interpolación lineal de la función PWL; b) Interpolación lineal del rango de la función PWL

debe respetar la restricción que la suma de los pesos sea unitaria:

$$w = \sum_{l=1}^{m+1} \mu_l v_l^k, \qquad v_l^k \in \widetilde{V}_D^k,$$

donde $\mu_l \in \Re^1$, $l = 1, \cdots, m+1$ y $\sum_{l=1}^{m+1} \mu_l = 1$ (Fig. 3.5 b)).

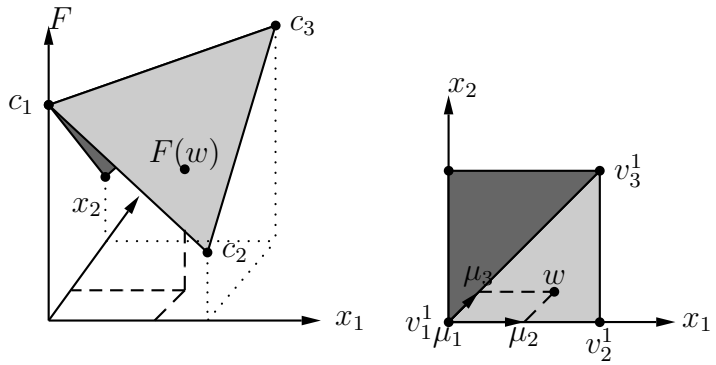
Paso 3 : Se debe computar el valor de la salida a partir de los valores de la función de transición en los vértices del símplice. Para ello se realiza la suma de estos valores de salida pesados por los mismos coeficientes que se encontraron en el *Paso 2* al describir la entrada como combinación convexa de los vértices del símplice.

$$F(w) = \sum_{l=1}^{m+1} \mu_l c_l.$$

donde c_l, $l = 1, \cdots, m+1$ son los valores de la función en los vértices $\widetilde{V}_D^k$ y μ_l es el peso correspondiente al vértice v_l^k (Fig. 3.5 a) y b)).

3.4.1. Evaluación temporal de la función de transición

Una manera simple de calcular el valor de una función utilizando este algoritmo consiste en desarrollar los pasos en forma secuencial a lo largo del tiempo. De esta



a) b)

Figura 3.6: a) Interpolación de la función a partir de los valores de la función en los vértices (c_1, c_2 y c_3); b) Identificación del vector w dentro de su símplice ($\widetilde{V}_D^1 = \{v_1^1, v_2^1, v_3^1\}$) junto con los pesos de su descomposición como combinación convexa de los vértices (μ_1, μ_2 y μ_3)

manera, se logra una estructura de cómputo muy eficiente y compacta, que además posee una realización sencilla utilizando circuitos electrónicos. Dado un dominio simplicial determinado por una partición regular en pasos δ en cada una de sus coordenadas, los vértices de cada símplice están dados por:

$$\begin{aligned} V_D = \quad & \{\, \mathbf{v} \in \Re^m : v_i = p_j \times \delta; \\ & i = 1, \cdots, m;\ p_j \in Z, 0 \leq p_j \leq p, p \times \delta = 1\}. \end{aligned} \tag{3.13}$$

A su vez, el conjunto de símplices queda determinado de la siguiente manera:

$$\begin{aligned} \widetilde{V}_D = \Big\{ & \widetilde{V}_D^k = \mathbf{v}^r + \{\mathbf{v}^1, \cdots, \mathbf{v}^i, \cdots, \mathbf{v}^{m+1}\} : k = 1, \cdots, m! \times p^m; \\ & \mathbf{v}^r \in V_D; \quad \mathbf{v}_j^1 = 0, j = 1, \cdots, m; \quad \mathbf{v}_j^{m+1} = 1, j = p, \cdots, m;\ \ s \\ & \|\mathbf{v}^{i+1} - \mathbf{v}^i\| = p,\ i = 1, \cdots, m\} ., \end{aligned} \tag{3.14}$$

donde $\widetilde{V}_D^k$ es el símplice k-ésimo, descripto por el conjunto de vértices que lo forman, $\mathbf{v}^r$ es un vértice base de $\widetilde{V}_D^k$, m es la dimensión del espacio y p el número de divisiones en cada dimensión provocado por la partición δ (ver Fig. 3.7).

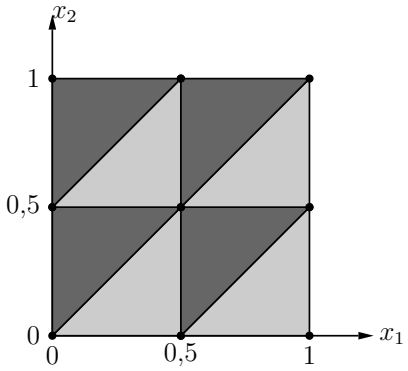


Figura 3.7: Dominio simplicial de partición regular $\delta = 2$

La evaluación temporal se realiza en sincronismo con una señal temporal del tipo rampa (ver Fig. 3.8). Conforme esta señal va evolucionando, y de acuerdo al valor de entrada presente, se van identificando secuencialmente los vértices del símplice que contiene la entrada. Simultáneamente se obtiene el peso con el que cada vértice contribuye a la combinación convexa. La identificación de cada vértice se obtiene de la comparación del valor de la rampa (equivalente al tiempo total transcurrido) con el valor de cada componente del vector de entrada. El resultado de esta comparación produce a lo largo del tiempo una palabra digital que varía, identificando instante a instante los vértices del símplice. La fracción de tiempo que cada vértice aparece en la palabra digital resultado de la comparación, a lo largo de un ciclo de la rampa representa el peso de la combinación convexa [5]. La rampa finaliza cuando alcanza el valor máximo de la entrada.

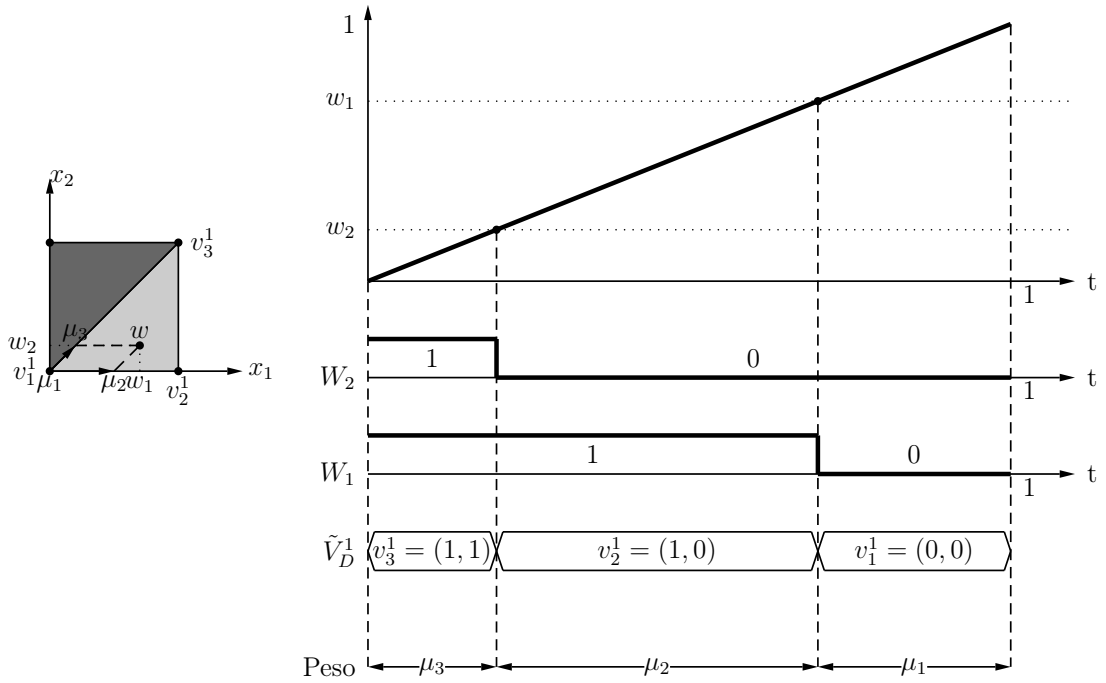


Figura 3.8: Codificación temporal de la entrada y obtención de los pesos de la combinación convexa

3.4.2. Implementación en señal mixta

La realización analógica del método propuesto supone que las entradas y los estados de las celdas en la esfera de influencia son señales analógicas y acotadas en un rango dado de tensión. Las señales se convierten utilizando una codificación temporal y esto se realiza con la ayuda de una rampa de tensión. La rampa auxiliar evoluciona desde su valor inicial hasta su valor máximo y con el uso de un comparador por entrada se determina el tiempo preciso en que la rampa sobrepasa a cada entrada. De esta forma, el valor de tensión de cada entrada se convierte en una señal binaria que comienza en uno al inicio de la rampa y conmuta a cero cuando el comparador detecta el cambio de magnitudes.

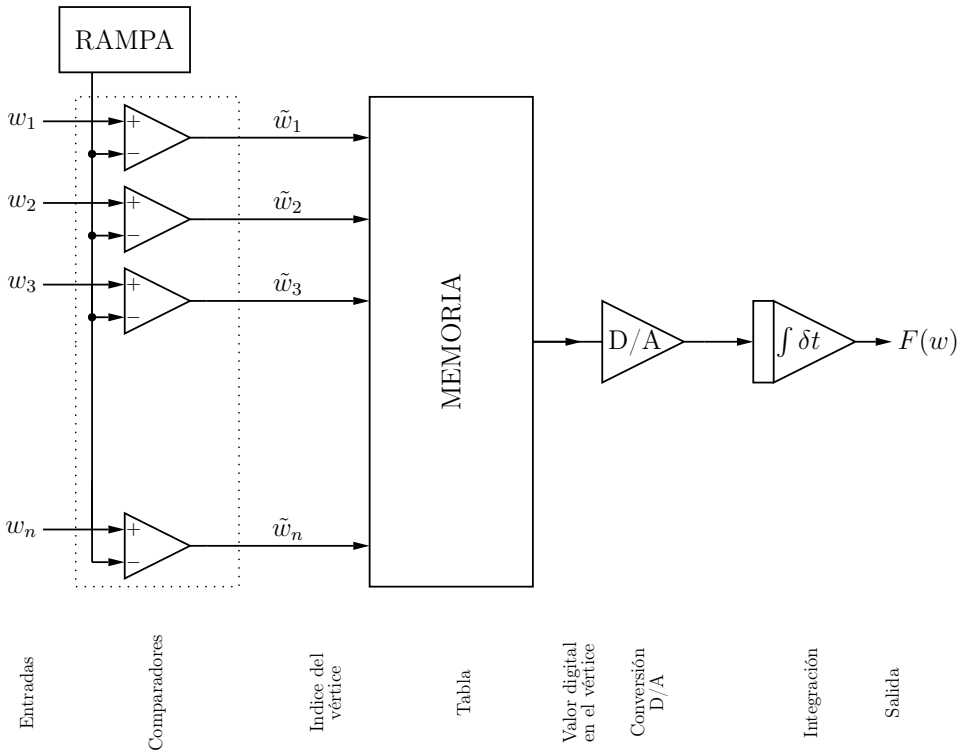


Figura 3.9: Diagrama en bloques de la celda S-CNN de señal mixta

Durante la ejecución de la rampa se produce un conjunto de señales binarias, correspondientes a cada una de las entradas, que agrupadas forman una palabra

digital. Esta palabra identifica a lo largo del tiempo los diferentes vértices que determinan el símplice; comenzando con el vértice opuesto al origen. Cuando la rampa sobrepasa el valor de una de las componentes de la entrada, la componente del vértice asociada cambia de uno a cero, y se anuncia el siguiente vértice. Al concluir la rampa, la palabra digital termina en el vértice de origen. El tiempo que cada vértice resultó expuesto (luego de normalizar la duración de los intervalos con la duración de la rampa) determina el conjunto de coeficientes en la combinación convexa de cada vértice. Resta entonces utilizar estos pesos para producir la salida. Esta tarea se puede realizar en paralelo con la descomposición, mediante la integración temporal de las contribuciones de cada vértice. Para ello se utiliza la palabra digital de entradas y estados como índice a una memoria (tabla de búsqueda) que produce como salida el valor de la función en el vértice. Este valor selecciona una fuente de corriente programable que suministra corriente a un integrador. Al finalizar la rampa la tensión de salida del circuito integrador es proporcional al valor de la función que se deseaba calcular. En la Fig. 3.9 se muestra una representación en bloques de la celda de señal mixta.

3.4.3. Implementación digital

La realización puramente digital de la estructura propuesta supone que todas las variables en uso, incluido el tiempo, están discretizadas. Las señales de entrada y el estado de cada celda se encuentran almacenadas y poseen una longitud de palabra fija. El tiempo es la variable con la que el método computa un nuevo estado, y se encuentra discretizado en intervalos iguales con la misma longitud de palabra que el estado de la celda. En la implementación digital se dispone de un contador ascendente que comienza con el ciclo de evaluación. En cada paso de cuenta se compara su valor con las entradas y los estados. La salida de estas comparaciones son las señales de un bit que contienen la información de las entradas y estados codificadas en tiempo (U_{Pwm} y X_{Pwm}). Agrupando estas señales, entradas y estados respectivamente, se tiene una palabra digital de nueve bits (para una esfera de influencia de tres elementos por tres elementos) que varía a lo largo de la cuenta del contador y que identifica el vértice a computar durante esa cuenta. Si se acumulan las

salidas de la memoria, que estas señales direccionan paso a paso durante la cuenta, al final del ciclo se obtiene en el contador el nuevo valor del estado. En la Fig. 3.10 se muestra una representación en diagrama de bloque de la implementación digital de la celda.

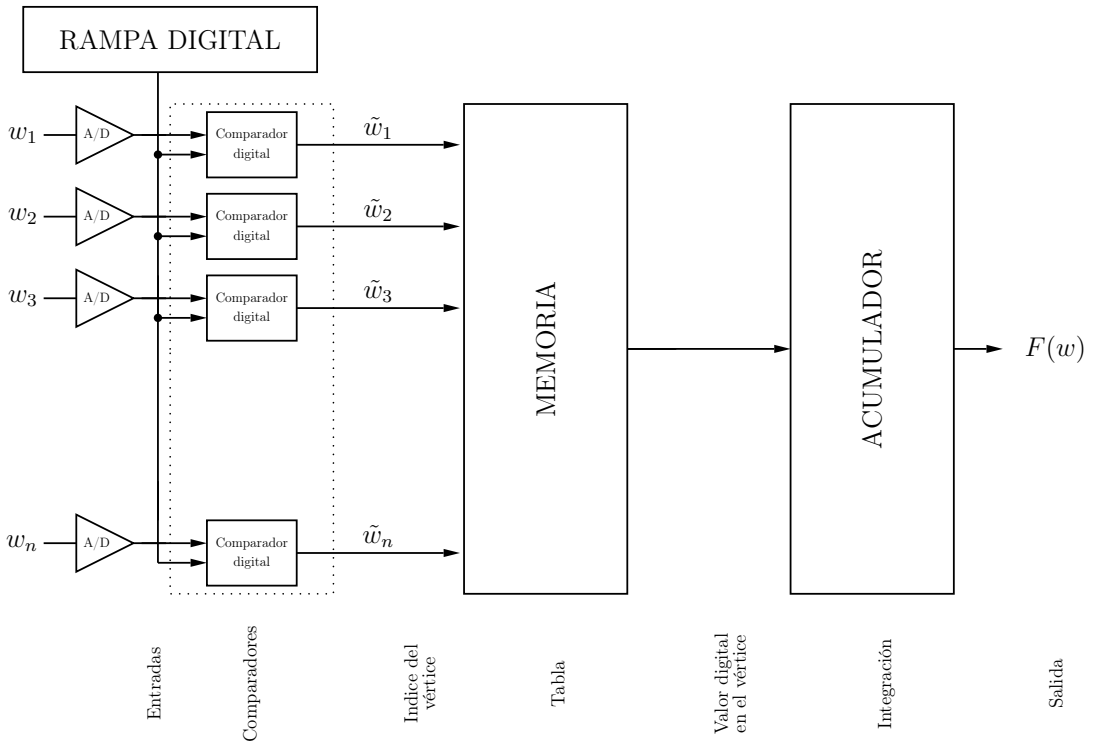


Figura 3.10: Diagrama en bloques de la celda S-CNN digital

3.4.4. Programación

En esta subsección se presentan los procedimientos básicos para programar un *imager* del tipo S-CNN. El método de programación se desarrolla para una esfera de influencia radio 1, (3×3 celdas). Los procedimientos son idénticos para otras configuraciones de la vecindad, tanto en tamaño como en forma. En una CNN estándar la obtención de un programa es un proceso de prueba y error complejo dado que es necesario encontrar la ecuación diferencial que converge al resultado deseado sobre la imagen [9]. En el caso de la S-CNN las tareas se obtienen en forma directa,

por construcción. Para ello, se examina el conjunto de posibles pares entrada salida y en base a esto, se construye la tabla de verdad que las describe. De hecho, para programar el *imager* S-CNN se necesita solamente identificar la relación entre un estado de la celda en $t = k + 1$ con las entradas y los estados de las celdas en su esfera de influencia en el instante $t = k$. El dominio de este mapeo funcional es el espacio de entrada-salida de la S-CNN . Por ser $F(\cdot)$ (3.12) una función PWL, se halla definida completamente por los valores que toma la función en los vértices. Por lo tanto, para identificar la función se enumeran todos los vértices, es decir se enumeran todas las combinaciones posibles de u y de x que pertenezcan a un vértice y se asigna el estado deseado como resultado de la celda obteniendo una tabla como la siguiente:

$t = k$	$t = k + 1$
$\{u_{i-1,i-1}(k), \ldots, u_{i+1,i+1}(k),$ $x_{i-1,i-1}(k), \ldots, x_{i+1,i+1}(k)\}$	$x_{i,i}(k+1)$
$\{0, 0, \ldots, 0\}$	c_0
$\{0, 0, \ldots, 1\}$	c_1
$\ldots$	.
$\{1, 1, \ldots, 1\}$	c_{511}

En esta tabla, los valores de $x_{i,i}(k+1)$ son directamente los valores de la función en los vértices (c). En casos donde la función dependa solamente de la entrada, o de la salida, es posible reducir la extensión de la tabla. Igualmente, en ciertos casos puede ser posible formular directamente la expresión lógica de la función, sin recurrir a la tabla de verdad de la misma.

Programas más complejos pueden obtenerse usando una combinación de programas de un solo paso y/o cargando los estados iniciales o las condiciones especiales de la frontera. Sin embargo, en todos los casos la filosofía de programación es la misma: se diseña un programa de un paso cuya aplicación en forma repetida converja al estado estacionario deseado.

3.5. Conclusiones

En este capítulo se han presentado los conceptos básicos de las redes celulares no lineales CNN. En particular se han detallado dos redes: en primer lugar, la CNN estándar, por ser la primer estructura reportada, y además porque debido a su simplicidad ha originado una gran cantidad de aplicaciones y desarrollos; en segundo lugar, la S-CNN que establece la arquitectura que se utilizará para el desarrollo de esta tesis. También se ha mostrado un método para el cálculo digital de la función PWL de la red S-CNN en forma distribuida a lo largo de un cierto número de ciclos de reloj. La estructura resultante es muy sencilla, y solo requiere en cada celda la implementación de un comparador y la disponibilidad de acceso a una memoria. Dado que la memoria es la misma para todas las celdas, es posible colocarla fuera de la misma -lo cual se explicará en detalle en los próximos capítulos. De esta manera, se cuenta con una estructura factible de implementar y con la capacidad para obtener *imagers* S-CNN en tecnología de circuitos integrados. Es también importante destacar que la estructura propuesta permite implementar redes CNN generales, sin la limitación que tienen las realizaciones existentes (basadas en la estructura CNN estándar propuesta por Chua) que solo permiten implementar funciones lógicas separables. Por otro lado, la programación de la red S-CNN puede obtenerse directamente por inspección de la tabla de verdad de la función de un paso deseada, con lo cual se evita el proceso de prueba y error requerido para el diseño de una CNN estándar.

Capítulo 4

Arquitectura del Imager CNN Simplicial

4.1. Introducción

Un imager con procesamiento S-CNN es un circuito integrado que puede capturar imágenes (cámara fotográfica) y que posee la capacidad de procesar en el plano focal de acuerdo a la mecánica de procesamiento de una CNN simplicial. El imager está formado por un arreglo de células o celdas, llamadas píxel, que tienen la capacidad de medir intensidad de luz, almacenar distintos valores y realizar el procesamiento teniendo en cuenta la vecindad del píxel. Las características de las celdas como la cantidad de registros que la componen, el tipo de información a almacenar, el tamaño y la forma de la vecindad, las funciones que se desean aplicar, la forma de guardar y obtener los valores de los parámetros de la función y la forma de correr el algoritmo de procesamiento, varían según las tareas que se requieran cumplir. Todas estas características hacen de la celda una arquitectura muy versátil, con varios grados de libertad, que pueden manipularse para implementar una celda más pequeña, más rápida o con diferentes capacidades de procesamiento.

Además del arreglo de píxeles (celdas) se pueden agregar otros bloques que realicen tareas específicas o dedicadas, y que sirvan para aumentar la capacidad de procesamiento del circuito. Estos bloques extras dan soporte a la estructura y ejecutan algoritmos de procesamiento y análisis que el arreglo CNN no puede hacer, como

calcular el código cadena de una imagen o realizar la correlación de dos cadenas de datos. La capacidad de procesamiento del sistema, dependerá del diseño de la celda y de las estructuras extras colocadas, siendo la selección de la arquitectura un paso fundamental en el diseño del imager. En este capítulo se describe la arquitectura general de un sistema de procesamiento S-CNN. Partiendo del algoritmo explicado en el capítulo anterior, se explican los bloques circuitales que realizan los pasos necesarios, se detallan las tareas que deben hacer cada uno de los bloques y las diferentes formas de realizarlas. Por otro lado, se analiza el impacto en el desempeño del circuito cuando se varían los factores de diseño, como por ejemplo el tipo de vecindad, el tamaño de los registros, etc. Para ello, se analizan diferentes combinaciones y se mide su impacto en el tamaño de la celda, el tiempo de operación, la capacidad de procesamiento, el consumo de potencia, etc.

4.2. Operaciones básicas de las celdas

Cada una de las celdas del arreglo debe ejecutar una serie de tareas (Fig. 4.1) como adquisición de imágenes, conversión analógica a digital, procesamiento de la información y envío de los datos al exterior, las cuales se describen a continuación.

- Adquisición de imágenes y conversión a valores digitales
  Cada pixel debe tener la capacidad de medir la intensidad de luz que incide en él. Para esto cada una de las celdas cuenta con un dispositivo de transducción de luz que se encarga de obtener un valor de tensión proporcional a la intensidad de luz. En caso de que se trabaje en forma digital, el valor de tensión que entrega el transductor debe ser convertido a un valor digital para ser almacenado en un registro y utilizado a posteriori como entrada de la función.

- Procesamiento
  Una vez almacenado el valor de la entrada en el registro, se procede a realizar el cálculo del próximo estado o de la función CNN que va a ejecutar la celda, teniendo en cuenta los valores de la propia celda y de los vecinos. Para realizar el procesamiento de la imagen almacenada es necesario seguir el algoritmo explicado en el capítulo anterior, el cual consta de una rampa de K pasos,

donde los valores de la entrada y el estado son comparados con este valor. Los valores resultantes de la comparación de todos los vecinos forman la dirección de memoria donde se almacenan los valores de las funciones. A ambos valores se los opera con una función lógica y este valor se integra en un acumulador. Al finalizar la *rampa de programa*, el acumulador tiene almacenado el valor procesado. Para esto la celda debe estar comunicada con sus vecinos, debe tener acceso a una memoria donde se almacena la función a calcular, y debe tener la lógica necesaria para cumplir con todos los pasos del algoritmo.

- Carga y lectura de imágenes

  Cada celda debe tener un banco de registros al cual se pueda acceder desde el exterior tanto para leer los valores de los registros como para cargar datos. Además de poder cargar los datos del transductor y del acumulador, se deben poder cargar datos desde el exterior para precargar el arreglo con imágenes, patrones o máscaras.

Adquisición de imágenes	Medir intensidad de luz
	Digitalizar
Almacenar imágenes	Guardar valor de digitalización
	Cargar datos desde el exterior
	Leer los datos
Procesar imágenes	Obtener la señal pwm
	Obtener los valores de la función
	Operar los valores de la función
	Integrar el valor de la operación

Figura 4.1: Tareas que debe realizar una celda S-CNN.

En resumen cada celda debe tener los circuitos necesarios para realizar las siguientes operaciones:

- Medir intensidad de luz.
- Convertir el valor de intensidad de luz (entrada de señal) en un valor digital.
- Almacenar la entrada de señal.
- Almacenar uno o más valores de estados.
- Comparar el estado de la celda y la entrada con la *rampa de programa*.
- Almacenar el valor de la señal PWM.
- Enviar y recibir la señal PWM hacia y desde los vecinos.
- Obtener el valor de la función en el vértice.
- Integrar el valor de la función.
- Habilitar los registros para lectura y escritura.

4.3. Estructura de la celda

Como se mostró en la sección anterior cada una de las celdas del arreglo debe ser capaz de realizar varias tareas. En esta sección se detallan los bloques necesarios para poder realizarlas. Las estructuras se muestran en la Fig. 4.2, donde el bloque adquisidor y convertidor A/D mide la intensidad de luz y la convierte en un valor digital, el banco de registros almacena todos los valores necesarios de la celda, y el motor de procesamiento PWL realiza las tareas necesarias para ejecutar el algoritmo de procesamiento.

4.3.1. Bloque adquisidor de imagen

Este bloque mide la intensidad de luz y la convierte en un valor digital. El bloque adquisidor de imágenes debe constar de un sensor de intensidad de luz y una

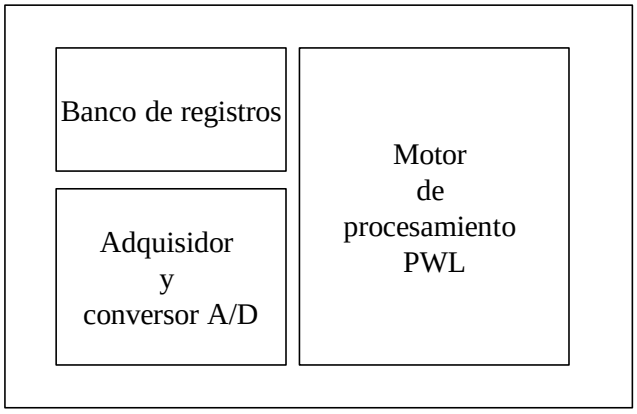


Figura 4.2: Diagrama en bloques de la celda.

estructura que digitalice su valor. Los dos sensores típicos para medir intensidad de luz en tecnología CMOS son el fotodiodo y el fotogate.

Un fotodiodo puede construirse de maneras diferentes en un proceso estándar CMOS. En el caso de un proceso con sustrato p existen dos alternativas. La primera alternativa consiste en una juntura p-n implantando una difusión n de alto nivel de dopado (comúnmente denominada n+) directamente en el sustrato p; la otra alternativa consiste en una juntura p-n entre el sustrato implantado n (ó n-Well) y el sustrato p de la oblea de silicio o wafer. Al incidir fotones en el silicio, estos crean pares hueco-electrón. Para separar el hueco del electrón y crear una corriente eléctrica, se utiliza un campo eléctrico que acelera las cargas. En este caso, se utiliza la existencia de campo eléctrico dentro de la región de vaciamiento de una juntura p-n polarizada en inversa. La corriente generada por la incidencia de los fotones descarga la capacidad de la juntura p-n. Luego de un tiempo de integración, la tensión en los terminales del diodo será inversamente proporcional a la intensidad de luz.

Los fotogates son otra de las posibilidades para realizar un sensor de luz, a través de una estructura formada por polisilicio, aislante y material semiconductor. Esta estructura implementa un capacitor MOS que al estar expuesto a la luz e intercambiar carga con un circuito externo se convierte en un fotogate. El voltaje de compuerta se elige para que el capacitor trabaje en la zona de vaciamiento de forma

que los portadores mayoritarios del semiconductor (sustrato) se alejen de la superficie formando una región de vaciamiento. En dicha región, al igual que en los fotodiodos, los pares hueco-electrón generados por la absorción de luz son recolectados.
Una gran diferencia del fotogate con respecto al fotodiodo es que los pares hueco-electrón no se suman en la generación de una corriente sino que son integrados en la carga contenida en el capacitor MOS. Esta alternativa de fotodetector tiene como principal ventaja un menor nivel de ruido. Por otro lado la presencia del gate en la zona expuesta a la luz incidente reduce la eficiencia cuántica, en particular para las ondas de mayor energía (o menores longitudes de onda) que son absorbidas en gran medida por el polisicio del gate.

Existen dos formas de utilización de los sensores de luz: lectura directa de la corriente o de integración de la corriente en un capacitor.
El método de lectura directa de corriente es utilizado principalmente en aquellos casos que se necesita que la cámara trabaje en tiempo continuo, y consiste en la medición de la corriente a través del mismo. Este método en general tiene un nivel de ruido mucho mayor que el de integración y se utiliza solo en los sistemas donde su característica de tiempo continuo lo justifique. El método de integración de la corriente consiste en cargar el dispositivo en el momento de inicio de la toma de la imagen y luego dejarlo a circuito abierto de forma que la corriente generada por la luz incidente vaya descargando la capacidad parásita del dispositivo. Al cabo de un tiempo de espera, o tiempo de integración, se tiene la medida representada por la carga residual en el fotodiodo (o photogate) lo que es equivalente a la tensión del dispositivo. Este modo de uso es de tiempo muestreado y variando el tiempo de exposición se puede variar la ganancia de conversión del dispositivo. Para leer la carga residual en el dispositivo se pueden utilizar dos configuraciones llamadas pasiva y activa.

La arquitectura con fotodetector pasiva llamada PPS (en ingles: Passive Pixel Sensor), está formada por un diodo fotoconversor polarizado en inversa y un transistor funcionando como llave analógica que lo vincula a la línea de lectura. La 4.3 muestra el esquema de fotodetector pasivo. La carga del fotodetector a una tensión de inicio se hace por medio de la línea de lectura.

La arquitectura con fotodetector activo o APS (por sus siglas en inglés, active

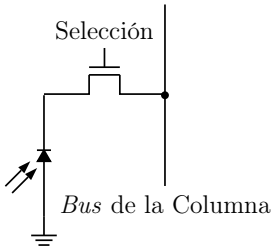


Figura 4.3: Esquemático de un fotodetector pasivo.

pixel sensor), está constituida por un fotodiodo, un amplificador en configuración de seguidor para no cargar el fotodiodo, y transistores de paso y de reinicio. Estos últimos son necesarios para el acceso a la línea de lectura. En la Fig. 4.4 se muestra el fotodiodo que integra la carga para realizar la fotoconversión, seguido de un transistor NMOS que funciona como amplificador seguidor de fuente. Se puede observar que para inicializar el diodo se utiliza un transistor NMOS, dado que si se utiliza un PMOS el pixel resultaria más grande debido al NWELL del transistor.

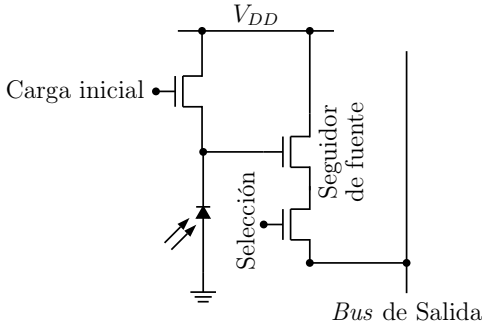


Figura 4.4: Esquematico de un fotodetector activo.

La conversión a valores digitales de la tensión muestreada se puede realizar de varias formas. Se puede colocar un solo convertidor analógico-digital (A/D) rápido con la precisión deseada fuera del arreglo, y realizar la conversión de las celdas una por una, o se puede colocar un convertidor A/D simple y pequeño en cada una de las celdas. Si se elige colocar un solo convertidor A/D fuera del arreglo, los tipos de conversores más utilizados son de aproximaciones sucesivas, Flash o Sigma-delta

dependiendo de la velocidad y de la precisión. Si se coloca el conversor en cada una de las celdas, éste debe ser lo más pequeño posible para minimizar el tamaño total del arreglo. Por este motivo el convertidor de rampa simple es el más indicado, dado que para realizar la digitalización de la señal muestreada, solo se necesita un comparador que realice la comparación entre la señal del sensor de luz y una rampa externa. La salida del comparador luego se utiliza como habilitación de un registro en el cual se guarda el valor de la conversión.

4.3.2. Banco de Registros

En el banco de registros se almacenan los datos de la digitalización del transductor de luz, y otros valores de estados. Los registros pueden estar formados por un arreglo de Flip Flops, Latches, o un arreglo de transistores. Deben tener habilitaciones y multiplexores para ser seleccionados, tanto para escritura como para lectura. La máxima precisión de los registros depende del nivel de ruido del elemento transductor de luz. La cantidad de registros mínima es de dos, uno para almacenar el valor de la entrada, y otro para almacenar el valor del estado. Si se colocan más registros, se puede tener mayor cantidad de estados intermedios o registros auxiliares. La habilitación de los registros para la escritura puede ser externa o interna. Algunos registros se pueden habilitar cuando suceden eventos particulares como por ejemplo si un registro es menor que otro, o si un registro es menor que el valor del contador, o si un registro es menor que el valor del BUS, etc. Las formas de habilitar los registros dan al sistema diferentes capacidades de procesamiento.

A modo de ejemplo, en el caso que se desee calcular el flujo óptico, el valor de la celda debe ser un vector que apunte al vecino más parecido. Es decir, se debe comparar la celda con cada uno de los pixeles de la imagen siguiente, encontrar el menor y guardar su dirección. Para esto es necesario en un primer paso, almacenar en un registro la diferencia de intensidades y en otro registro la dirección. En un segundo paso, se compara con otro pixel, y si la diferencia es menor, las celdas donde se almacenan la diferencia y la dirección, se deben habilitar para almacenar los nuevos valores. Esta pequeña modificación se logra agregando la salida del comparador a la habilitación de los registros.

4.3.3. Motor de procesamiento PWL

El motor de procesamiento PWL es el encargado de ejecutar el algoritmo explicado en el capítulo anterior. Este bloque contiene un comparador, para comparar el estado y la entrada de la celda con la *rampa de programa*, las conexiones con los vecinos para enviar y recibir la señal PWM, la lógica que permite obtener el valor de la función en el vértice (leer el dato de la memoria) y un acumulador para integrar el valor de la función. La celda también puede poseer registros para almacenar valores intermedios, dependiendo del tipo de arquitectura particular seleccionada. Los bloques básicos y su interconexión se muestran en la Fig. 4.5.

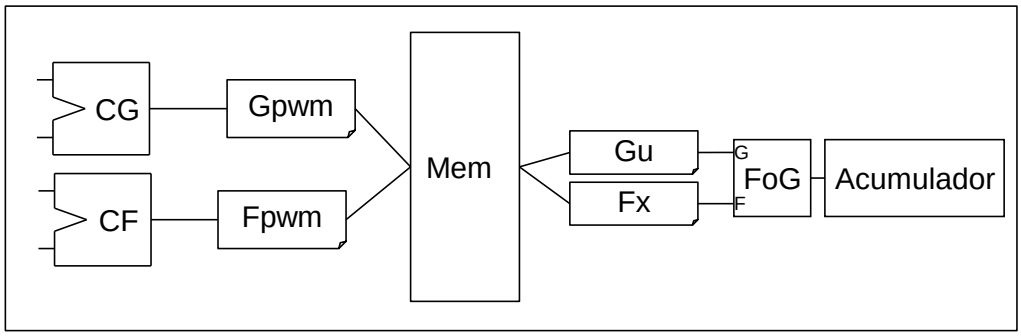


Figura 4.5: Bloques del motor de procesamiento PWL.

Los comparadores CF y CG comparan la señal de entrada *rampa de programa* con el valor en los registros U y X. Sus salidas, las señales Fpwm y Gpwm, se envían a todos los vecinos a la vez que se reciben las señales correspondientes de todos los vecinos. Estos vectores entran a un bloque que obtiene el valor de la función en el vértice de la memoria. Los valores leídos de la memoria, Gu y Fx, ingresan al bloque FoG donde son operados lógicamente. La salida del bloque FoG ingresa a un acumulador que va integrando los valores sucesivos.

El bloque encargado de obtener el dato de la memoria se puede realizar de distintas maneras. Su arquitectura varía en función de la ubicación de la memoria (dentro o fuera de la celda). Si la celda tiene acceso a toda la memoria este bloque está formado por un multiplexor que habilita la salida del dato apuntado. Si la memoria no se encuentra en la celda y es compartida por las celdas del arreglo, hay

dos formas de acceder a ella: de a una celda por vez, mediante un acceso aleatorio; o conectando todas las celdas a la memoria, y transmitiendo dirección por dirección el contenido de memoria de manera que cada celda recoja el valor de memoria que le corresponde. El análisis de las distintas configuraciones de memorias y las formas de leerla se describen en la Sección 4.5.2

4.4. Estructuras extras

Además del arreglo de celdas interconectadas, el imager puede dotarse con estructuras extras que le den soporte. Estas estructuras se pueden agregar para facilitar y automatizar el manejo del imager, ejecutando automáticamente el algoritmo de procesamiento, o para agregar mayor capacidad de procesamiento. En esta sección se muestran algunos bloques que sirven para aumentar la capacidad de procesamiento, y así poder realizar tareas más complejas y extender las capacidades de la red CNN. Estos bloques pueden separarse en tres grupos: los que realizan operaciones matemáticas específicas, los que ejecutan un algoritmo o una secuencia de pasos, y los bloques de soporte.

4.4.1. Operaciones matemáticas específicas

Un primer tipo de bloques a considerar son los que realizan operaciones matemáticas simples como sumas, restas, multiplicación, etc. La utilidad de estos bloques radica en la posibilidad de calcular valores de descriptores de imagen, y generar valores que luego se puedan utilizar en un análisis estadístico para reconocimiento. Una lista de los bloques que realizan operaciones matematicas especificas se describe a continuación.

- Sumador/Restador: La estructura básica más simple y más utilizada son los sumadores, dado que es muy usual realizar estos cálculos en sistemas de procesamiento. Se pueden colocar varias de estas estructuras para que realicen cálculos en paralelo, y se comporten como un sistema SIMD, o también se pueden colocar sumadores de diferente precisión para trabajar con valores numéricos (obtenidos de un análisis de imagen) más grandes o precisos, que sirvan

para realizar operaciones de más alto nivel.

- Contador: La función principal de un bloque contador en el arreglo es realizar la suma total de los valores de todas las celdas en el arreglo, de manera de contar cuantos píxeles hay en la imagen, y la intensidad que tienen. Se puede utilizar para calcular valor medio de la imagen, la de contar la cantidad de píxeles, calcular áreas, perímetros, y así obtener factores de forma para poder identificar objetos o separar en regiones.

- Integrador: Un integrador es un circuito que se va contando y guardando el valor de la cuenta en un acumulador, de manera de ir integrando los valores que ingresan en él. Este circuito tiene una entrada y una salida, donde la salida es de mayor cantidad de bits que la entrada, de manera de poder contar varios valores. Si se comparten recursos como los sumadores, y se desea saber el valor de la suma total, el integrador se utiliza para ir guardando los valores de las sumas parciales.

- Multiplicador y Divisor: Se pueden agregar al sistema bloques que realizan operaciones específicas de multiplicación y división. Cada uno de estos bloques con dos entradas y una salida del doble de precisión de los valores de entrada. Este bloque se puede utilizar para calcular muchos de los descriptores de imágenes. Para obtener descriptores que sean invariantes ante el tamaño del objeto o la rotación hay que multiplicarlos o dividirlos por valores calculados.

4.4.2. Ejecución de algoritmos

Un segudo tipo de bloques son aquellos que ejecutan una serie de tareas específicas. Estas tareas pueden variar dependiendo de las necesidades o de las características del sistema. Los algoritmos que ejecutan estos bloques pueden ser de control, entre los que se incluyen el manejo de un arreglo de celdas, la generación de señales para realizar la conversión A/D, o también cálculos más complejos como transformadas de valores de la imagen o cálculos estadísticos para la detección de objetos.

- Bloque de procesamiento automatizado CNN: Este bloque tiene como finalidad generar todas las señales necesarias para hacer funcionar al arreglo de celdas CNN. Básicamente es una máquina de estados con una interfaz al usuario de una entrada (Comenzar) y una salida (Finalizado). Al darle una señal de "*Comenzar*", este bloque corre el algoritmo de procesamiento de la CNN, generando automáticamente las señales de habilitación de los bloques de las celdas, y la *rampa de programa*.

- Codificador código cadena: Este bloque se encarga de calcular la codificación código cadena de una imagen, que puede ser el esqueleto o el borde de un objeto. Este bloque debe contar con configuraciones específicas como el lugar y la dirección de comienzo del recorrido, o la acción a tomar en caso que suceda una bifurcación de la imagen. También debe contar con un arreglo de registros que almacene el valor de los píxeles visitados.

- Correlador: El correlador calcula la correlación (la suma de las diferencias) entre dos cadenas de datos. Como entrada tiene dos arreglos de registros, y la salida es la suma de las diferencias entre cada uno de los elementos particulares. Este bloque se puede utilizar para comparar códigos cadena (asumiendo una normalización previa). Si no se conoce el comienzo del recorrido, uno de los registros de entrada, se puede ir rotando, para ir calculando la correlación entre cada una de las posiciones. Este bloque también se puede utilizar para comparar un conjunto de descriptores de imagen y poder realizar un reconocimiento de patrones o imágenes comparando los descriptores con una base de datos.

- Microprocesador: Los algoritmos que realizan un procesamiento de bajo, medio y alto nivel, muchas veces constan de varios pasos deben ser ejecutados en un orden especifico, para obtener la información deseada. Además, un sistema complejo que consta de varios bloques, como los mencionados anteriormente, debe ser controlado, de una manera eficaz para obtener resultados satisfactorios. Un microprocesador que controle todos los bloques es necesario para interconectarlos controlarlos y correr los algoritmo de procesamiento y análisis

de imágenes.

4.4.3. Soporte

- Memoria Externa: La memoria externa se utiliza para guardar datos externos necesarios para realizar el procesamiento y análisis de imágenes. Entre ellos se pueden nombrar, imágenes patrones de comparación, estado inicial del arreglo para realizar el procesamiento, funciones para la CNN, lista de valores de descriptores, valores de códigos cadena de objetos, lista de procesos que correrá la CNN, el programa que correrá el microprocesador, almacenar valores intermedios de procesos, comunicar los datos con el exterior, etc. Todos estos valores pueden estar disponibles fuera del chip o en una memoria interna.

- Bloque Control de la CNN: Como se ha visto, el control del arreglo de celdas puede estar a cargo de una unidad especializada o de un microprocesador, pero también muchas veces es necesario poder ingresar señales desde el exterior. Para poder controlar el imager desde el exterior para realizar tareas especificas diferentes a las estandares, como por ejemplo en la binarización, donde no es necesario correr una rampa completa, o en casos de segmentación donde la *rampa de programa* no crece monótonamente (ej: Suma ponderada, caso código cadena). Este bloque selecciona la fuente de señales de la entrada al arreglo. En resumen, este bloque selecciona quien tiene el control del imager.

- Comunicaciones: La idea de un sistema de visión es procesar y analizar las imágenes para extraer información. Esta información se debe transmitir al exterior de alguna forma, para comunicar los resultados, compartir datos con otros circuitos para coordinar entre varios una tarea o un análisis en conjunto o para tomar algún tipo de decisión a más alto nivel. Varios sistemas de comunicación se pueden colocar en los circuitos. A continuación se nombran algunos dependiendo de la aplicación específica y del ambiente donde trabajará el circuito:

    - UART: Se puede agregar al sistema una unidad asincrónica de recepción

y transmisión, para transmitir información con el protocolo de comunicaciones RS232. Una gran cantidad de dispositivos se comunican a través, de este protocolo, de esta manera el chip se podría comunicar con la PC, con un sistema de transmisión de datos por Radio Frecuencia, etc.

- SPI: La colocación de un bloque de Interfaz de Comunicación serie con periféricos, permite comunicar al sistema con una gran cantidad de dispositivos, desde una placa de video, una memoria tipo SD, un dispositivo Bluethoot, etc. También sirve para intercomunicar varios chips para que trabajen en paralelo.
- PARALELO: Una salida y entrada de bus del tipo paralelo se puede utilizar para intercomunicar varios circuitos integrados, para enviar imágenes, instrucciones, datos, configuraciones, información procesada, etc, de un chip a otro. La comunicación entre chips logra que varios de estos sistemas de visión trabajen en conjunto logrando obtener una mayor calidad de la información obtenida, ya que la información se puede complementar, utilizando un buen algoritmo de fusión de datos.
- I^2C: Es un protocolo muy usado para comunicar micro controladores con periféricos y también para la comunicación de varios chips trabajando sobre la misma placa.

4.5. Variaciones en la arquitectura

Existen diferentes factores de diseño que juegan un rol significativo en la arquitectura e impactan de diversas maneras en el desempeño del arreglo. Algunos de estos factores son: la selección de la vecindad, el método usado para obtener el valor de la función en el vértice, la cantidad y el tamaño de los registros y la opción de compartir algunos de los recursos de las celdas. Algunos de los índices de desempeño utilizados para evaluar los factores de diseño son: el tamaño de las celdas, el tiempo de procesamiento, la complejidad de ruteo, la capacidad de procesamiento, el fill factor, el consumo, etc.

El tamaño de las celdas impacta directamente en el tamaño del arreglo que se

puede lograr. A menor tamaño de la celda mayor será el arreglo que se logrará por unidad de área y mayor la resolución de la imagen. El tiempo de procesamiento impacta en la velocidad con la que se puede obtener información de una imagen, y directamente con la cantidad de información que se puede extraer por unidad de tiempo. La complejidad de ruteo impacta en la dificultad de interconectar todos los bloques en el chip, haciendo más compleja la creación de la máscara o generando mayores restricciones en el programa de place & route. La capacidad de procesamiento indica que clase de información se puede extraer de la imagen.

También hay otros marcadores que como el factor de relleno (en inglés: Fill Factor) que indica la relación entre el área de adquisición y el área total del pixel. En un imager que solo detecta imágenes el fill factor es del orden del 35 %, si este valor es muy chico, indica que el sensor es muy chico en relación al tamaño de la celda, por lo tanto se pierde mucha área que se puede aprovechar para adquirir luz. También el consumo en algunos casos puede llegar a ser crítico o importante, sobre todo en dispositivos móviles en los que se debe alimentar al circuito con una batería o pilas.

En esta sección se muestra cualitativamente el impacto de los factores de diseño (diferentes configuraciones de celdas) en las capacidades de procesamiento indicando si son mayores o menores teniendo en cuenta las diferentes configuraciones estudiadas.Se realiza un análisis del impacto que tienen cada uno de los diferentes factores de diseño en el desempeño de cada una de las celdas y del arreglo. Primero se describe la selección de la vecindad, luego la forma de distribuir la memoria a las celdas, luego la selección de los registros y por último se analiza la opción de compartir recursos.

4.5.1. Selección de vecindad

Cada una de las celdas está conectada a otras celdas y esta interconexión es llamada vecindad. La entrada de n elementos de la celda, depende de la cantidad de vecinos con la que la celda está conectada. Si la celda está conectada con cuatro vecinos, n será cinco, ya que son 4 vecinos y la celda. La conexión con los vecinos puede realizarse de varias formas diferentes, tal como lo muestra la figura 4.6. En

un arreglo ortogonal, se pueden seleccionar los 4 vecinos más cercanos en forma de cruz (a), los 4 más cercanos en forma de equis (b), los 8 que la rodean (c), los 24 con un radio de 2 vecinos, etc. También se puede realizar un arreglo hexagonal, y conectarse con los 7 vecinos (d), 18 vecinos, etc.

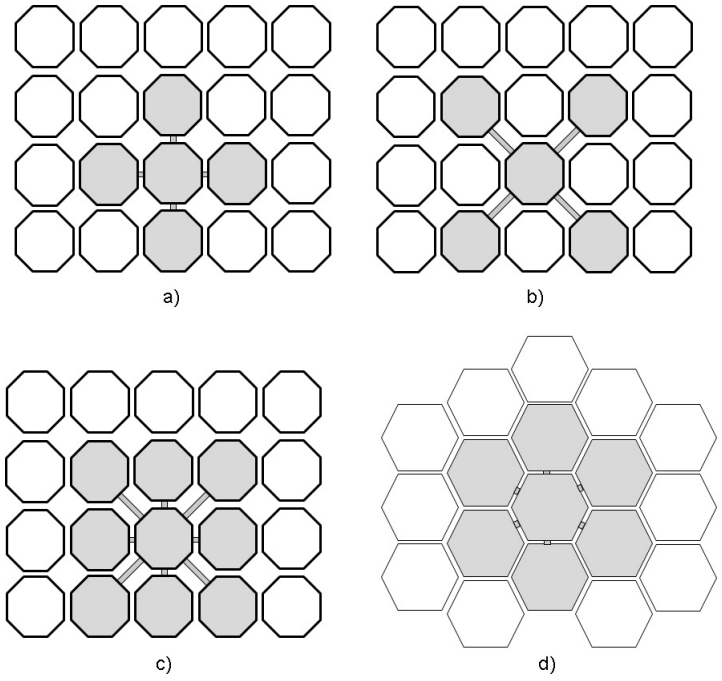


Figura 4.6: Conexiones con vecinos.

Dado que se cuenta con dos funciones *(F y G)* que se pueden aplicar a diferentes registros y operar con diferentes funciones lógicas, se pueden combinar estas operaciones para ampliar la capacidad de procesamiento del arreglo. Se puede modificar el radio de vecindad de cada celda, realizando la iteración de varias funciones y operándolas lógicamente.

1. Aumentar el radio de vecindad: Si se tiene la misma imagen en dos registros, se le pueden aplicar funciones iterativas a una imagen (para mantener el pixel central) e ir trasladando la copia, cambiando la función a aplicar, de manera de aumentar el radio de vecindad cuya función final es la combinación de las funciones aplicadas luego de la traslación.

2. Cambiar la forma de la vecindad: Si se tiene dos vecindades complementarias, se puede lograr una vecindad mayor (la suma de las dos vecindades) aplicando una función a cada vecindad, operándolas con el producto, e iterando hasta cubrir todas las combinaciones posibles.

3. Combinando la 1 y la 2 y con las funciones adecuadas, se puede aumentar y cambiar la forma de la región de vecindad y, por ejemplo aplicar eficazmente un algoritmo de AOF (explicado en el capítulo 2) teniendo en cuenta regiones más grandes que las abarcadas por la vecindad.

Existe la posibilidad de que la entrada de una de las funciones en vez de provenir de los vecinos, provenga de un registro interno de la celda. Así los vectores *Vf o Vg* , en vez de provenir de la concatenación de los vecinos, se pueda obtener de un valor almacenado en la celda. De esta manera, y utilizando la operación AND, se pueden aplicar funciones selectivamente a distintas celdas, utilizando este registro como una máscara.

Selección de la vecindad:

Tamaño de la celda: La cantidad de vecinos puede impactar en el tamaño de la celda si la memoria se distribuye completa a la celda. En este caso cada celda debe tener los multiplexores para leer el dato del vértice y, como el tamaño de la memoria aumenta exponencialmente con la cantidad de vecinos, también aumentan en tamaño los multiplexores. Caso contrario, si la memoria se envía de a un bit a la vez, no habría impacto alguno en el tamaño de la celda.

Tiempo de procesamiento: La cantidad de vecinos puede impactar en el tiempo de procesamiento de la celda, si la memoria se transmite a todas las celdas en serie, ya que el tamaño de la memoria aumenta exponencialmente con la cantidad de vecinos y el tiempo para recorrerla seria mayor. Caso contrario si la memoria se distribuye completa a todas las celdas, no habría impacto en el tiempo de procesamiento.

Complejidad de ruteo: Al aumentar la conectividad con los vecinos, aumenta la cantidad de cables, o líneas de metal, que van de celda a celda y aumenta también exponencialmente la cantidad de líneas de memoria que hay que llevar a las celdas.

Al haber más líneas de metal que colocar en la máscara, la complejidad se incrementa y la dificultad para llevar a cabo el ruteo aumenta.

Capacidad de procesamiento: Al estar conectado con una mayor cantidad de vecinos la capacidad de procesamiento aumenta, dado que las funciones que se pueden realizar son más complejas y la cantidad de variables para tener en cuenta a la hora de tomar decisiones son mayores. Las capacidades de procesamiento y las funciones posibles a implementar varían con el tipo de vecindad seleccionada. Por ejemplo es imposible detectar líneas diagonales con una vecindad en forma de cruz (+).

Memoria: El tamaño de la memoria se incrementa exponencialmente al aumentar la cantidad de vecinos, ya que la cantidad de datos a almacenar en la memoria es de $M = 2^n$, donde n indica el tamaño de la vecindad.

4.5.2. Distribución de los valores de la función

La memoria donde se almacenan los vértices debe estar accesible a todas las celdas. Como se mencionó previamente, esta puede ser colocada en cada una de las celdas o fuera del arreglo. Dado que colocar una memoria en cada celda produciría una redundancia de estructuras, y que cada memoria tiene los mismos valores, esto haría la celda innecesariamente grande. Por esta razón, es conveniente colocar una sola memoria fuera del arreglo de celdas, de manera que todas las celdas puedan leer su valor. Vale recordar que todas las celdas ejecutan la misma función por lo tanto el valor de la memoria es el mismo para todas las celdas, pero los valores a recolectar dependen del valor de la celda y de los vecinos, así que los datos que cada celda lee son de la misma memoria pero no es el mismo dato. Como se aclaró previamente esta tarea puede ser realizada de diferentes maneras:

1. Cada celda accede a la memoria: El acceso a la memoria es organizado separando a las celdas por turnos, y en cada ciclo de reloj, una celda distinta realiza la lectura del valor correspondiente de la memoria. Esto significa que el tiempo de lectura, de la memoria, del arreglo completo depende de la cantidad de celdas. Si el arreglo tiene C Columnas y F Filas, la lectura de la memoria, toma FxC ciclos de reloj. En este caso por cada uno de los K pasos de la *rampa de programa*, se debe realizar toda la secuencia de lectura de la memoria,

de FxC ciclos de reloj. El tiempo total de procesamiento en este caso será de F x C x K pasos. Siendo el tiempo total de procesamiento proporcional a la precisión de los registros de la celda (K) y al tamaño del arreglo (FxC)

2. Se envía la información de la memoria a todas las celdas: Esta tarea puede realizarse de dos maneras distintas.

   a) *Los valores de la memoria son enviados en forma serial, utilizando una línea accesible por todas las celdas*: En este caso los datos se envían de a uno, comenzando desde la dirección más baja hasta el último. Se indica, a través del bus de entrada, la dirección de la memoria, y colocando en la línea de memoria el valor correspondiente a la dirección. Cada celda almacena en un registro interno, en el momento adecuado (cuando la dirección de la memoria corresponde a su vector PWM), el valor que le corresponde. Este proceso toma $M = 2^n$ pasos de reloj. Un comparador de tamaño n y un registro deben ser agregados a la celda, para poder comparar la dirección de la memoria a leer con la enviada por la memoria y para almacenar el valor. En este caso por cada uno de los K pasos de la *rampa de programa*, se debe realizar todo el envío de la memoria, que llevara M ciclos de reloj. El tiempo total de procesamiento en este caso será de $M \times K$ pasos. Siendo el tiempo total de procesamiento proporcional a la precisión de los registros de la celda (K) y al tamaño de la vecindad (M).

   b) *Los valores de la memoria son enviados en forma paralela:* En este caso los ($M = 2^n$ valores) son enviados todos al mismo tiempo, a todas las celdas, donde cada celda, con un multiplexor, selecciona el valor que corresponde. En este caso en un ciclo de reloj, todas las celdas seleccionan la línea de memoria que corresponde y leen el valor al mismo tiempo. Siendo el tiempo total de procesamiento solamente proporcional a la precisión de los registros de la celda (K).

Si se comparan los dos casos, el paralelo es más rápido, porque en un solo ciclo de reloj todas las celdas leen su dato de la memoria. En este caso, un multiplexor del

tamaño de la memoria (M) debe ser colocado en cada celda. El caso serie es más lento, y es necesario colocar un comparador y un registro, pero la celda no posee multiplexor (que es una estructura que puede ocupar casi el 30 % de la celda). Combinando estos dos casos, se puede encontrar una configuración óptima en tamaño. En vez de mandar una línea o todas se pueden enviar algunas líneas, y multiplexarlas tanto en la celda como en tiempo. Por cada grupo de líneas que se retiren de la distribución de la memoria, el mutiplexor de la celda se debe hacer más chico, y se debe agregar un bit más al comparador de direcciones. La tabla 4.1 muestra un análisis de tamaños y tiempos usando 1, 2, 4, 8 y 16 líneas de memoria, para una vecindad de 4 y la tabla 4.2, usando 1, 2, 4, 8, 16, 32, 64, 128, 256 y 512 líneas para una vecindad de 8. Como se puede ver en la Fig. 4.7, el tamaño mínimo de celda se logra enviando 16 líneas. Este no es el caso más rápido, pero es el que logra la celda más pequeña.

	8 Vecinos			
LINEAS	Area Mux	Area Comp	Area Total	Ciclos
1	0	99	99	512
2	6	88	94	256
4	10	77	87	128
8	20	66	86	64
16	30	55	85	32
32	60	44	104	16
64	120	33	153	8
128	240	22	262	4
256	480	11	491	2
512	960	0	960	1

Tabla 4.1: Tamaños y cantidad de ciclos necesarios para procesar una imagen, dependiendo de las lineas utilizadas para transmitir la memoria, para una vecindad de 9 pixeles.

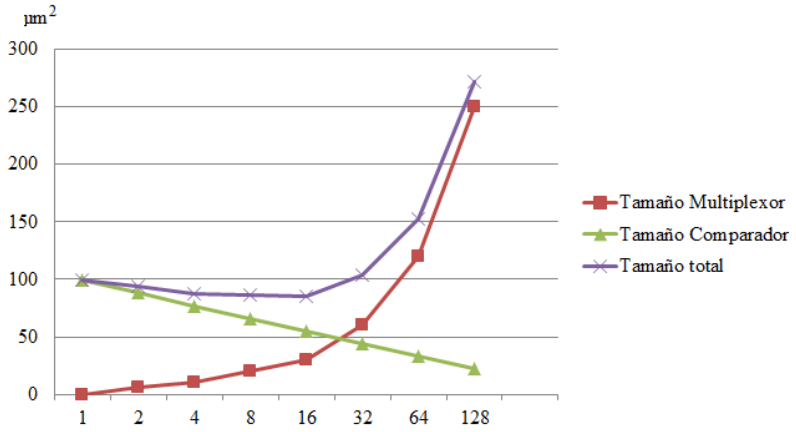


Figura 4.7: Curva tamaño de celda en funcion de bits de memoria enviados.

La memoria puede ser de la precisión que se desee. En este caso los valores posibles de los vértices son de un bit. Si la memoria es de más de un bit, la cantidad de líneas aumenta proporcional a la cantidad de bits de palabra de la memoria. Este análisis es independiente del tamaño de los registros, ya que solo depende de la conectividad con los vecinos y con el tipo de metodología utilizada para realizar la lectura de la memoria.

Impacto de la forma de distribuir la memoria:

Tamaño de la celda: El tamaño de la celda depende de la configuración utilizada para leer la memoria. Dado que al enviar menos líneas el comparador aumenta su tamaño, y al enviar más líneas el multiplexor aumenta su tamaño. El tamaño mínimo de celda se logra enviando 16 líneas; si se envían más o menos bits, la celda incrementa su tamaño.

Tiempo de procesamiento: El tiempo de procesamiento depende altamente de la configuración de lectura de memoria utilizada. Al distribuir mayor cantidad de líneas, menor es el tiempo de procesamiento.

LINEAS	4 Vecinos			
	Area Mux	Area Comp	Area Total	Ciclos
1	0	55	55	32
2	6	44	50	16
4	10	33	43	8
8	20	22	42	4
16	30	11	41	2
32	60	0	60	1

Tabla 4.2: Tamaños y cantidad de ciclos necesarios para procesar una imagen, dependiendo de las lineas utilizadas para transmitir la memoria, para una vecindad de 5 pixeles.

Complejidad de ruteo: Si la memoria se encuentra fuera del arreglo, hay que distribuirla a todas las celdas, cuantas más líneas de memoria se envíen a las celdas, mayor será la complejidad en el ruteo.

Capacidad de procesamiento: La distribución de la memoria no tiene impacto alguno en la capacidad de procesamiento de la celda.

Memoria: El tamaño de la memoria no depende de la forma con la que se distribuyen los datos.

4.5.3. Registros

Cada celda posee registros para almacenar información. Como fue mencionado antes, un registro es utilizado para almacenar el valor de la entrada (la imagen obtenida del sensor de luz) y otros registros son utilizados para almacenar estados de la celda. Las modificaciones que se pueden realizar con respecto a los registros son dos. Se puede variar la cantidad de registros y la precisión de cada uno (la cantidad de bits)

Cantidad de Registros

El número de registros (P), depende de la cantidad de información que sea necesario guardar en cada celda para realizar tareas de cierta complejidad. Más registros pueden ser agregados a la celda para obtener estados intermedios, para almacenar mayor cantidad de imágenes, o para tener mayor cantidad de estados. Por ejemplo: Si se desea separar objeto de fondo, y obtener bordes, un estado seria borde, y otro estado seria fondo, y esos valores, se guardan en dos registros diferentes. Si se desea aplicar un filtro gaussiano, solo 2 registros son necesarios, la imagen de entrada y el registro destino donde se guardara la imagen filtrada. Si se desea ver la diferencia entre dos imágenes, 3 registros son necesarios, Las dos imágenes a comparar y el registro destino. En el caso de realizar el seguimiento de cada píxel para ver como es el movimiento de la imagen (calcular el flujo óptico), se deben poder almacenar las dos imágenes a comparar, se necesita un registro para ir almacenando el valor del píxel más similar y otro para almacenar la dirección de ese píxel.

Cantidad de registros:

Tamaño de la celda: Al aumentar la cantidad de registros aumenta el tamaño de la celda. Como mínimo cada celda debe tener 2 registros, uno para la imagen obtenida (entrada), y otro para la imagen almacenada (estado). En un circuito con 2 registros de una precisión de 8 bits, al agregar un registro más la celda aumenta casi un 10 % su tamaño.

Tiempo de procesamiento: La cantidad de registros no tiene ningún impacto en el tiempo de procesamiento.

Complejidad de ruteo: Al aumentar la cantidad de registros, se deben colocar más multiplexores y lógica de habilitación para poder escribir y leer en ellos. Esto no hace mucho más complejo el ruteo.

Capacidad de procesamiento: Al aumentar la cantidad de registros, la cantidad de información con la que se cuenta en cada celda es mayor, por lo tanto las funciones que se pueden realizar son más.

Memoria: El tamaño de la memoria no depende de la cantidad de registros.

Precisión de los registros

La cantidad de bits de los registros, depende en la aplicación específica del chip. La precisión puede variar desde 1 bit (Imágenes blanco y negras) hasta "q" bits. El máximo valor de "q" en el registro de entrada es función del ruido del píxel o de la presición del conversor A/D. En los registros de los estados, es función de los valores que puede tomar la función de salida. Por ejemplo, en nuestro caso los valores de la función solo pueden tomar 0 o 1, pero si la función que se le aplica a las celdas, puede tomar cualquier valor, teniendo en cuenta que el valor máximo del registro procesado será $q * b$, donde "q" es la precisión del registro y "b" es la cantidad de bits de la función.

La longitud de la rampa (K) depende directamente de la precisión del registro al que se le aplica la función, tiene una longitud de $k = 2^q - 1$. Esto significa que cuanto más grande sea el registro, mayor cantidad de pasos tendrá la *rampa de programa.*

En tabla 4.3 se muestra el tamaño de la celda dependiendo de la cantidad y la precisión de los registros. Hay que tener en cuenta que si los registros son más grandes el acumulador usado como integrados debe también ser más grande. La Fig. 4.8 muestra gráficamente como varia el tamaño del banco de registros al variar la cantidad y la precisión de los registros.

La cantidad de ciclos que se necesitan para ejecutar una tarea, no solo depende de la precisión de los registros, sino también del método de lectura de la memoria seleccionado. La tabla 4.4, muestra el impacto en el tiempo de procesamiento, medido en ciclos de reloj, dependiendo de la cantidad de líneas de memoria que se envían a cada celda y la precisión de los registros.

Variación en la precisión de los registros

Tamaño de la celda: Al aumentar la precisión de los registros el tamaño de los registros aumenta, por lo tanto aumenta el tamaño de la celda. El aumento depende de la cantidad de registros, pero cada

Tiempo de procesamiento: Si se aumenta la precisión de los registros, aumenta el tiempo de procesamiento, ya que la *rampa de programa* debe tener K pasos, y $k = 2^q - 1$.

Bits	Registros			
	1	2	3	4
10	478	544	610	676
9	446	506	566	625
8	415	469	521	574
7	385	432	478	524
6	355	395	435	474
5	325	359	392	425
4	295	322	348	375
3	265	285	305	325
2	235	248	262	275
1	205	212	218	225

Tabla 4.3: Tabla de tamaños de celda en μm^2 en función de la cantidad y la precisión de los registros.

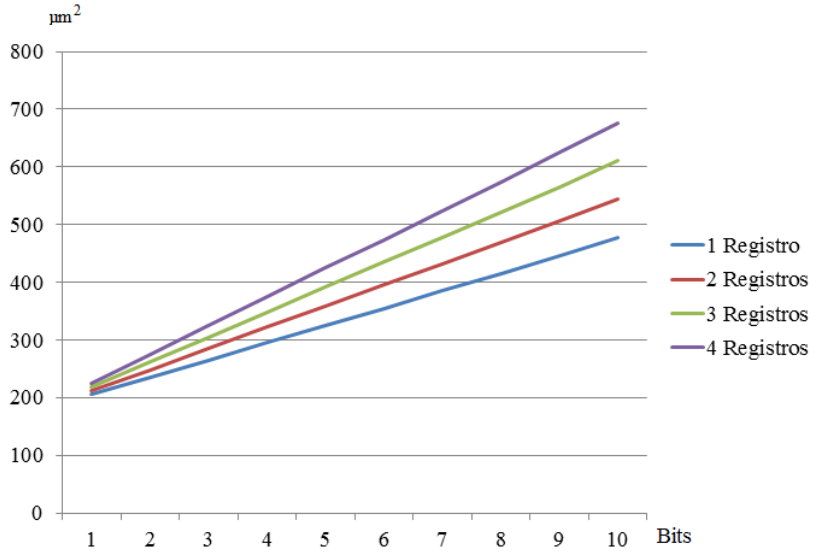


Figura 4.8: Tamaño del banco de registros en función de la precisión y la cantidad de registros.

Complejidad de ruteo: La complejidad de ruteo no aumenta con la precisión de los registros, solo aumenta el tamaño del bus.

Capacidad de procesamiento: Si se aumenta la precisión de los registros habrá más bits con los que trabajar, pero las funciones que se pueden aplicar son las mismas.

Memoria: El tamaño de la memoria no depende de la precisión de los registros.

4.5.4. Compartir Recursos

Para disminuir el tamaño de la celda se pueden compartir recursos, de manera de multiplexarlos en tiempo. En esta sección se muestran algunos bloques que pueden ser compartidos en un diseño CNN.

Bits	Cantidad de lineas					
	1	2	4	8	16	32
10	65474	32738	16370	8186	4094	2048
9	32706	16354	8178	4090	2046	1024
8	16322	8162	4082	2042	1022	512
7	8130	4066	2034	1018	510	256
6	4034	2018	1010	506	254	128
5	1986	994	498	250	126	64
4	962	482	242	122	62	32
3	450	226	114	58	30	16
2	194	98	50	26	14	8
1	66	34	18	10	6	4

Tabla 4.4: Cantidad de ciclos de procesamiento dependiendo de la cantidad de líneas de memoria que se envíen y de la precisión de los registros.

Recursos compartidos

La idea principal de compartir recursos es lograr una celda que tenga menos componentes y por ende, que sea más pequeña. Para lograr un funcionamiento equivalente, es preciso multiplexar en tiempo, es decir establecer turnos, para que las celdas utilicen el recurso. Hay una relación de compromiso entre tamaño y tiempo que en cada caso deberá ser analizada. El hecho de compartir recursos, no modifica la capacidad de procesamiento de las celdas, solo disminuye su tamaño a costa de tiempo de procesamiento. Hay recursos que se pueden compartir dentro de la misma celda, y otros que un grupo de celdas puede compartir. Dentro de una misma celda, se puede compartir los comparadores, y los multiplexores de lectura de memoria. Un grupo de celdas puede compartir el convertidor A/D, el motor de procesamiento PWM (comparador, buscador en emoria, integrador), etc. A continuación se detallan los recursos que se pueden compartir:

- Convertidor A/D: El convertidor A/D se puede compartir entre varias celdas o entre todas las celdas. Esta característica no modifica el tiempo de procesamiento, solo el tiempo que se tarda en adquirir las imágenes.

- Comparador para obtener palabra PWM: Sacrificando algunos ciclos de reloj, se puede utilizar un comparador para ser utilizado entre un grupo de celdas, o entre todas las celdas.

- Compartir Lógica para obtener valores de memoria: Si cada celda, almacena el valor la señal pwm de cada una de sus funciones, los multiplexores para leer de la memoria se pueden compartir entre varias celdas. Hay que tener en cuenta que cada celda debe tener el valor del vector pwm actualizado en toda su vecindad para realizar la búsqueda en memoria.

- Contador que integra el valor de la función: Si se desea compartir el contador, se debe agregar a cada celda un registro que guarde el valor del contador mientras lo está utilizando otra celda y actualizar el contador cuando le toque su turno.

Arreglo	Tecnologia						
	$0.5\mu m$	$0.35\mu m$	$0.18\mu m$	$0.13\mu m$	$90nm$	$60nm$	$45nm$
64×64	9.21	6.46	3.31	2.4	1.66	1.11	0.83
96×96	13.82	9.68	4.97	3.59	2.49	1.66	1.24
128×128	18.43	12.9	6.63	4.79	3.32	2.21	1.66
192×192	27.65	19.4	9.95	7.19	4.98	3.32	2.49
256×256	36.86	25.8	13.27	9.58	6.64	4.42	3.32

Tabla 4.5: Tabla de tamaños de arreglos en mm en función de la tecnología.

4.6. Tamaño del arreglo de celdas en función de la tecnología.

El tamaño de las celdas, y por ende del arreglo, depende de la tecnología que se esté utilizando. La tabla 4.5 y la Fig. 4.9 muestra, teniendo en cuenta un celda con dos registros de ocho bits, una vecindad de 4 vecinos, y con todos los valores de la memoria accesibles, como sería el tamaño de arreglos de diferentes tamaños en varias tecnologías.

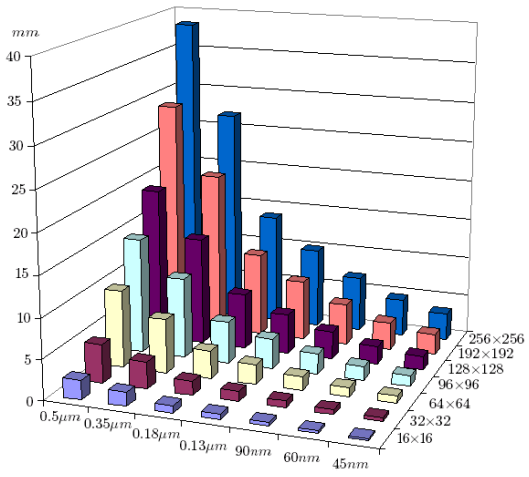


Figura 4.9: Tamaño del arreglo de celdas en función de la cantidad de celdas y de la tecnologia seleccionada.

4.7. Conclusión

En este capítulo se han mostrado distintos tipos de arquitecturas que son capaces de realizar un procesamiento del tipo S-CNN. En estas estructuras, hay varios aspectos que tiene un impacto en el tiempo de procesamiento y en el tamaño de la celda. La configuración más deseable sería una celda pequeña y con muchas capacidades de procesamiento y muy rápidas. Se ha mostrado que estas características están vinculadas, y al mejorar una característica, la otra se ve comprometida. Es trabajo del diseñador, seleccionar la arquitectura que mejor cumpla con los requisitos del procesador deseado.

Capítulo 5

Realización 2D

5.1. Introducción

En este capítulo se describe el desarrollo de un sensor de imágenes con capacidad de procesamiento S-CNN, fabricado en una tecnología CMOS de 90nm. Este imager consta de un arreglo ortogonal de 64×64 celdas con una región de vecindad de cinco celdas configurable, dos memorias de programa y sumadores de columna. Cada una de las celdas del arreglo posee cuatro registros, lo cual permite realizar varias tareas, como el cálculo de flujo óptico o la aplicación de funciones diferentes a píxeles distintos o regiones de píxeles distintas. Cada píxel contiene un sensor activo consistente en un fotodiodo y un circuito de adaptación de señal analógico. En cada celda se utiliza un comparador para realizar la conversión A/D de la señal de luz. La conversión se realiza con el método de rampa simple utilizando dos rampas, una analógica y otra digital, sincronizadas externamente. Para obtener una implementación eficiente en términos de área, se utilizó una única memoria en la periferia del chip y su contenido se distribuyó por completo a todas las celdas por medio de un bus de datos. Los sumadores de columna permiten realizar ciertas funciones de alto nivel sobre la imagen, como por ejemplo calcular la posición del centro de masa, los momentos de la imagen, etc.

El objetivo planteado para el diseño del imager fue lograr un circuito que implemente una CNN simplicial con mayor densidad de celdas y mayor velocidad que los reportados hasta el momento [15], y que estuviera en igualdad de condiciones con

procesadores de imágenes SIMD de funcionalidad similar [87, 73, 88]. En cuanto a la funcionalidad, se impuso la condición de que el CI pueda calcular el flujo óptico de la imagen, lo cual determina la cantidad mínima de registros a utilizar en cada celda.

El capítulo está organizado de la siguiente manera. En la Sección 5.2, se describe paso a paso el flujo de diseño seguido, los programas utilizados para desarrollar, sintetizar, simular, crear las máscaras y testear la arquitectura. En la sección 5.3 se explica en detalle el tipo de arquitectura seleccionada, las celdas del arreglo, las estructuras extras, la distribución de los bloques en el circuito y su interconexión. En la sección 5.5 se describen todas las señales que controlan el circuito, se explican sus funciones y se ilustra la ejecución paso a paso algunas de las tareas básicas del circuito. En esta sección también se muestran simulaciones y testeos realizados previos a la fabricación, así como también algunos resultados experimentales del chip fabricado. En la Sección 5.6 se resumen las características más sobresalientes del circuito y se presentan las conclusiones.

5.2. Diseño

Para diseñar el circuito integrado, se siguió un procedimiento secuencial, tal como muestra la Fig. 5.1. Se comenzó por la selección de la arquitectura de la celda, realizando el análisis de las posibles configuraciones tal como fuera explicado en el Cap. 4; el detalle de la misma se describe en la Sección 5.3. La celda seleccionada fue primeramente descripta en lenguaje VHDL (Very large scale integration Hardware Description Logic) y luego simulada. Se describió en principio un arreglo de 14×14 celdas, el cual se sintetizó con el software de desarrollo ISE® de Xilinx, y se implementó en una FPGA Spartan 3E para testear su correcto funcionamiento y capacidades de procesamiento. Para realizar este testeo, se utilizó una interfaz programada en MATLAB®, la cual se comunica a través del puerto serie de la computadora con un microprocesador PicoBlaze® sintetizado en la FPGA. El micro controlador recibe los comandos desde la PC y genera todas las señales para controlar el chip. Toda la información del hardware y software utilizado para el testeo, se encuentra en el Cap 7. Una vez verificado su correcto funcionamiento y

capacidades de procesamiento, con imágenes cargadas externamente, se utilizó esta estructura como referencia para la creación del esquemático y máscaras del circuito final. En esta etapa se simuló y verifico la parte digital del circuito. La parte analógica se simuló con SPICE.

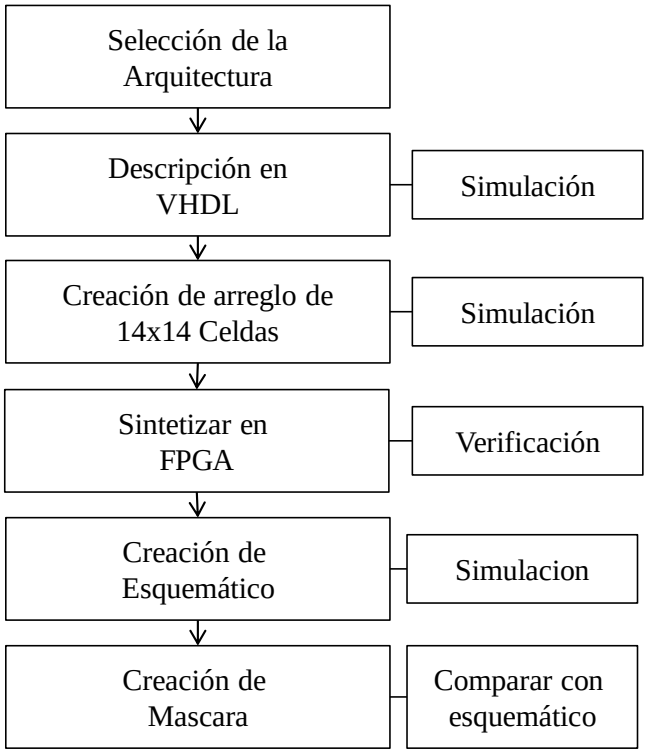


Figura 5.1: Diagrama de flujo del diseño seguido.

Para la creación de las máscaras, se utilizó un flujo de diseño Bottom-Up, es decir comenzando por la parte de más bajo nivel (transistores, compuertas básicas) y uniendo estas estructuras, se formó el sistema completo. Las máscaras se fueron organizando y acomodando para terminar con un píxel cuadrado de $25\mu m \times 25\mu m$. El software de diseño utilizado para la creación de los esquemáticos, las simulaciones y el diseño de las máscaras fue el paquete de software Cadence® y algunos detalles de la máscaras se realizaron en Tanner®. Primero se diseñó toda la parte digital, luego se dimensionaron y se crearon las máscaras de los buffers de las celdas, y una vez terminada la parte digital de la celda, se diseñó la parte analógica.

5.3. Arquitectura

Para la fabricación de este chip, se utilizó la tecnología CMOS de $90nm$ de la compañía UMC, de sustrato tipo P, con nueve niveles de metal y un nivel de polisilicio. El sistema completo consiste en un arreglo de 64×64 celdas donde la región de vecindad de cada celda es de cinco vecinos, seleccionable en forma de cruz ($+$) o equis ($\times$). Cada una de estas celdas dispone de registros auxiliares que permite la implementación de funciones complejas. Además de utilizar la información de la vecindad, se puede también utilizar el valor de un registro interno como dirección del vértice (de la función PWL) para aplicarle a la celda propiedades, o funciones particulares. La cantidad de niveles de metal disponibles en esta tecnología permite la distribución de todos los valores de memoria a la celda. Todas las celdas son accesibles a través de una dirección de fila y columna, tanto para su lectura como su escritura. Valores externos pueden ser grabados en todos los registros si se genera la habilitación adecuada. Se agregó una estructura adicional consistente en un sumador de columna con el objetivo de proveer más capacidad de procesamiento. La Fig. 5.2 muestra un diagrama en bloques del sistema completo.

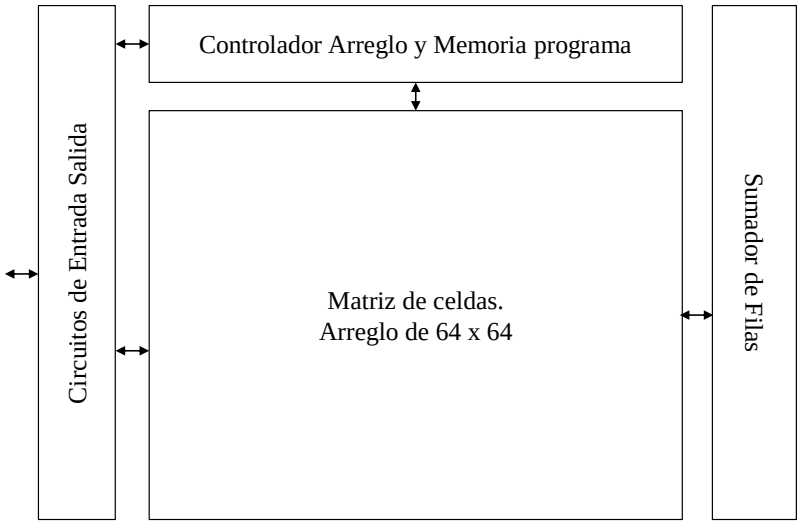


Figura 5.2: Diagrama en bloques del sistema.

El chip completo abarca un área de 2mmx2mm, y es capaz de funcionar con un

voltaje de trabajo entre 1V y 2V. La conexión con el exterior se realiza a través de 114 pads, en un encapsulado CQFP128. Las celdas lógicas digitales, las analógicas, los pads, y las estructuras de protección ESD fueron diseñadas, tanto a nivel esquemático como de máscara, especialmente a medida para este circuito

5.3.1. Selección de la arquitectura

La selección de la arquitectura tuvo como finalidad optimizar la velocidad (en cantidad de períodos de reloj) y dotar al circuito con la capacidad de realizar el cálculo de flujo óptico de las imágenes tomadas. La selección de la vecindad, la forma de distribuir la memoria a las celdas, la cantidad y la precisión de los registros fueron seleccionadas tal como se describe a continuación:

Selección de la vecindad: La selección de la vecindad tiene un impacto directo con el tamaño de la memoria e indirecto con el tiempo de procesamiento. Dado que para realizar el flujo óptico cada uno de los pixeles debe ser comparado con todos los vecinos de la imagen siguiente, cada pixel debe estar conectado con sus ocho vecinos, quedando así una vecindad de nueve elementos. Si se utilizaran nueve vecinos, la memoria de programa debería ser de $2^9 = 512$ bits de longitud, lo cual aumentaría la complejidad circuital y de ruteo. Debido a esto, en vez de conectar la celda con sus nueve vecinos, se separó la vecindad en dos grupos de cinco vecinos, quedando la posibilidad de conectar los cuatro vecinos a la celda central en forma de cruz (+) o equis (×), tal como fuera explicado en el Cap. 4. De esta forma se logra que cada celda esté conectada con sus nueve vecinos, y se disminuye el tamaño de la memoria de $2^9 = 512$ a $2^5 = 32$.

Distribución de la memoria: Para lograr una mayor velocidad de procesamiento se decidió enviar toda la memoria de programa a las celdas, de manera de asegurar la disponibilidad del valor de la memoria (valor de función en el vértice correspondiente) en la celda en un solo ciclo de reloj como se mostró en el Cap. 4 el mínimo tamaño de celda se logra enviando 16 señales de memoria a cada celda. En el caso bajo análisis se decidió enviar 32 señales en vez de 16, lo cual, si bien produce un aumento casi del 10 % en el área de la celda, al mismo tiempo produce un aumento del doble en la velocidad del circuito.

Registros: La cantidad de registros y la precisión tienen un impacto directo en el tamaño de la celda y en el tiempo de procesamiento, dado que aumentan el tamaño de los multiplexores, de los comparadores, del contador y de los buffers. A su vez, la rampa de programa debe ser más larga para recorrer todos los valores de los registros. Se decidió hacer los registros de las imágenes de seis bits dado que resulta suficiente para tareas de visión primaria y permite implementar un arreglo de 64×64 celdas en el área disponible de fabricación ($4mm^2$). Cada una de las celdas del arreglo posee cuatro registros, llamados X, U, W y T; tres ellos (X, U, y W) son de seis bits, y uno (T) es de 4 bits. Para calcular el flujo óptico de la imagen se necesitan almacenar dos imágenes sucesivas (que se almacenan en X y U) y en un tercer registro se debe ir guardando el menor valor de la comparación con los pixeles vecinos (se almacena en el registro W) junto con la dirección del pixel comparado (almacenado en el registro T). Dado que el registro T se utiliza para almacenar la dirección del pixel, no hace falta que sea de seis bits, dado que con cuatro bits es suficiente para codificar las nueve direcciones posibles de movimiento.

La distribución de la memoria, y la precisión de los registros hacen que el circuito procese como máximo en 128 ciclos de reloj y como mínimo en 2 dependiendo la función a realizar. Las características principales de la arquitectura se pueden resumir de la siguiente manera:

- Vecindad:
  La conectividad con los vecinos forma una vecindad de 5 celdas, seleccionable entre tipo cruz ($+$) o equis ($\times$).

- Cantidad y precisión de registros: En cada celda se han colocado cuatro registros, tres de seis bits, y uno de cuatro bits. La habilitación de los registros puede provenir del exterior, del conversor A/D o del comparador interno utilizado para calcular el valor de la señal PWM.

- Recursos compartidos:
  Toda la lógica necesaria para realizar el procesamiento de los estados se encuentra en la celda, es decir que no se comparte ningún recurso entre celdas.

El único recurso compartido entre todas las celdas es la memoria de programa. Cada celda tiene sus propios registros, motor de procesamiento PWL y estructuras de habilitación.

- Estructuras Extras:
  Como estructura extra se ha colocado un sumador de 11 bits por cada fila. Al seleccionar una columna, el sumador calcula el valor de la suma de los registros seleccionados (X, U, W o T) de todas las filas de la columna seleccionada.

- Condiciones de Borde:
  Las condiciones de borde pueden fijarse como "0" o "1" de manera externa (condición de borde fija).

- Función de transición:
  Las funciones que se pueden aplicar a los registros son dos, F y G, cuyo valor es de un bit. El valor de estas dos funciones puede ser operado con las funciones lógicas AND, OR o XOR.

- Memoria:
  Dado que la vecindad está definida por 5 celdas, la memoria debe ser de $2^5 = 32$ valores. En este caso cada uno de los valores de la memoria es de un bit, siendo cada una de las memorias de programa de 32 bits. La memoria está conformada por dos registros serie de 32 bits, ubicados fuera del arreglo y todos sus valores se distribuyen a todas las celdas, es decir que todas las celdas tienen acceso a los 32 valores cargados en la memoria en todo momento.

5.3.2. Bloques de la celda

Cada una de las celdas tiene los circuitos necesarios para almacenar y procesar información. Como se muestra en la Fig. 5.3 cada celda está compuesta por cuatro bloques. El *Adquisidor y Convertidor A/D*, que posee el sensor de fotones y los circuitos necesarios para convertir la intensidad de luz en una palabra digital de seis bits. El *Banco de registros*, que almacena las entradas y los estados de la celda. El *Motor de procesamiento PWL*, que posee un comparador, un acumulador y la lógica

de búsqueda en memoria necesarios para realizar el procesamiento descrito en el Cap. 3 y los *Buffers de salida* que habilitan la escritura de la celda en el BUS. A continuación se explica cada uno de los bloques en detalle.

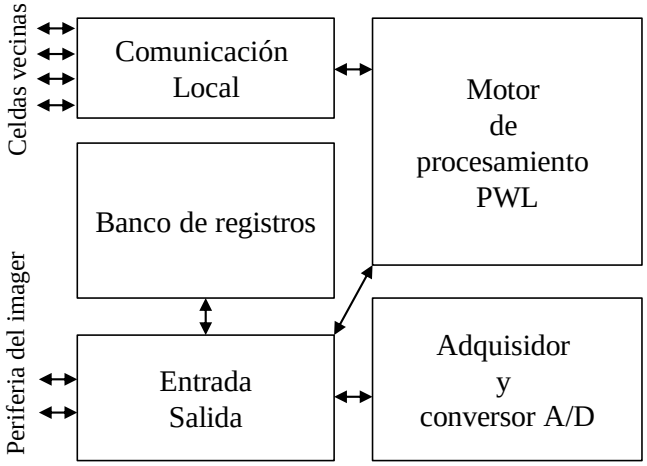


Figura 5.3: Diagrama en bloques de la celda del arreglo.

Adquisidor y conversor A/D

Se utiliza como elemento fotoconversor un sensor activo (APS) que incluye un fotodiodo realizado con una isla de implante n+ sobre el sustrato de tipo p. La Fig. 5.4 muestra el esquemático del APS donde se puede ver que el fotodiodo se conecta a la tensión de inicialización a través del transistor Mr. El transistor Mcs actúa como seguidor de señal para no cargar el diodo, y el transistor $Msample$, es utilizado para muestrear la señal y almacenarla en el capacitor $Chold$. El muestreo de la tensión se realiza en todas las celdas a la vez, consiguiendo de esta forma una toma instantánea tipo "snap shot". La Fig. 5.5 muestra la máscara de este circuito.

El conversor analógico digital utilizado en esta celda es un conversor denominado de rampa simple". Este mecanismo de conversión es bastante simple, utiliza pocos transistores y también permite realizar la conversión de todas las celdas en paralelo, compartiendo señales. El conversor posee un comparador que compara la señal de entrada, almacenada en el capacitor de muestreo, con una rampa analógica (creciente o decreciente según el caso) que evoluciona en sincronía con una rampa digital. Al

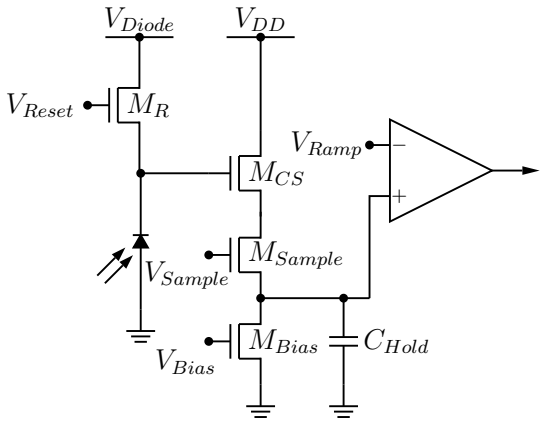


Figura 5.4: Esquemático del sensor activo.

inicio de la conversión el comparador habilita el almacenamiento de la información de la rampa en el registro destino (en este caso U). De esta manera, en el registro queda almacenado el último valor de la rampa digital, que coincide con el valor digital de la entrada. El comparador utilizado es del tipo conmutado. La comparación se realiza a través de la señal CLK una vez que las señales de entrada se encuentran debidamente establecidas y la salida del comparador ha sido equilibrada. La Fig. 5.6 muestra el esquemático del amplificador.

La señal Hab habilita el amplificador, polarizándolo y habilitando la salida. La señal CLK a través del transistor Mr cortocircuita los drenajes de los transistores, equilibrando las tensiones de salida. Al abrir el transistor Mr (colocando un "0" en la señal de CLK) los dos nodos se desequilibran de acuerdo a las tensiones de los transistores de entrada NMOS. Una de las entradas del comparador es la rampa analógica externa y la otra es la tensión almacenada en el capacitor $Chold$. En este caso, la salida está sincronizada con la señal de CLK. Para mejorar la señal de salida se utiliza un inversor, y en el caso en que el amplificador esté deshabilitado, a través de la señal Hab se fuerza un "0" a la salida. La Fig. 5.7 muestra la máscara de la celda completa. Se pueden ver en la misma el sensor de luz y el comparador. El tamaño del sensor es de $14\mu m^2$, el tamaño del comparador es de $12\mu m^2$, el tamaño del APS es de aproximadamente $25\mu m^2$ y el tamaño del bloque completo es de $40\mu m^2$.

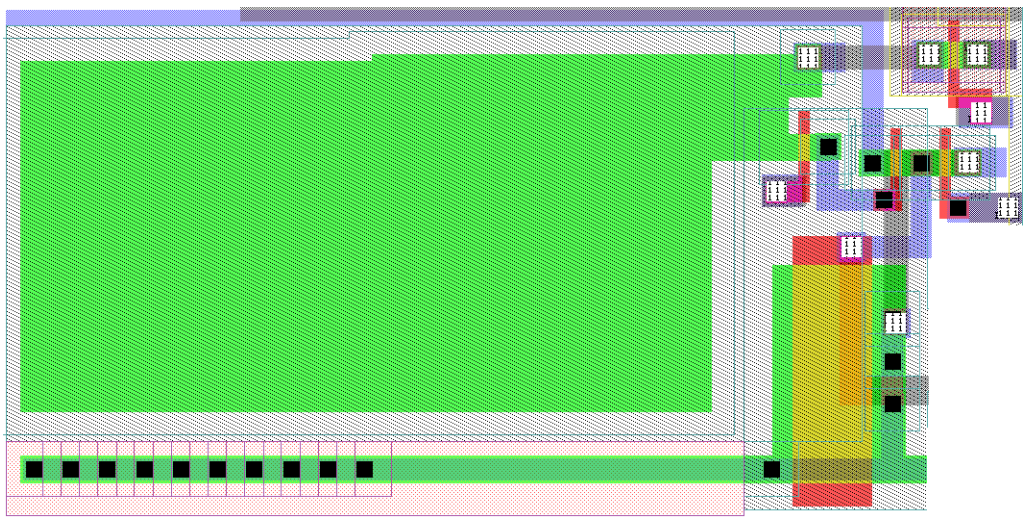


Figura 5.5: Layout del sensor activo.

Banco de registros

La celda cuenta con cuatro registros llamados X, U, W y T. Los registros X, U y W son de seis bits. En el registro U se almacena la imagen captada por los sensores de luz, mientras que los registros X y W se utilizan para guardar estados. El registro T es de 4 bits, y sirve para guardar imágenes de menor precisión como también para asignar diferentes tipos de "*banderas*" (flags) que puedan servir como una propiedad o característica asignada a un grupo de celdas y así poder dividir el arreglo en un subconjunto que posea la misma propiedad. El registro T se puede utilizar también como entrada de valor de vértice de la función G.

En este diseño, cada uno de los registros tiene una señal de reloj individual. La habilitación de los registros es diferente para cada uno, asignándole a cada registro propiedades de trabajo y operación distintas que permiten almacenar datos con distintas características y funciones. El dato en cada uno de los registros queda almacenado cuando se aplica un pulso en su señal de reloj mientras la señal de habilitación está activa. La Fig. 5.8 muestra un diagrama en bloques de los registros, sus habilitaciones y los multiplexores de lectura. Como se ve en la Fig. 5.8 cada registro tiene su propia señal de reloj; la habilitación de cada registro se describe a

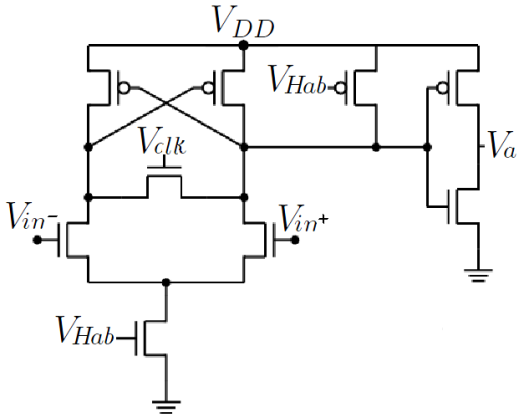


Figura 5.6: Esquemático del comparador.

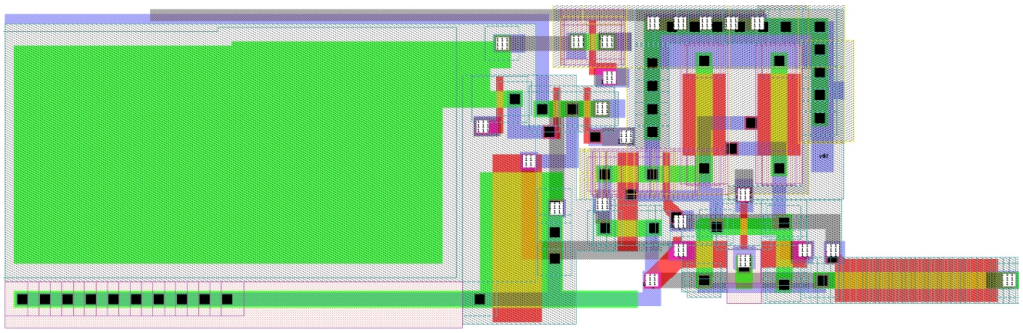


Figura 5.7: Máscara del sensor acivo con el comparador.

continuación y la tabla 5.1 muestra un resumen de las habilitaciones del registro U:

- Registro U: Este registro se puede habilitar de tres formas distintas, según se desee almacenar datos externos, almacenar el dato de la conversión A/D o guardar un dato ya procesado. Al seleccionar la celda a través de las señales COL y ROW, se habilita el registro para cargar un dato del exterior. Si se selecciona la salida del contador como entrada de datos al registro, se habilitan automáticamente los registros en todas las celdas del arreglo, y se almacenará en todas las celdas el valor del contador. Al realizar la conversión A/D, la habilitación de la celda está controlada por la salida del comparador, así cada celda puede almacenar el valor de intensidad de luz correspondiente.

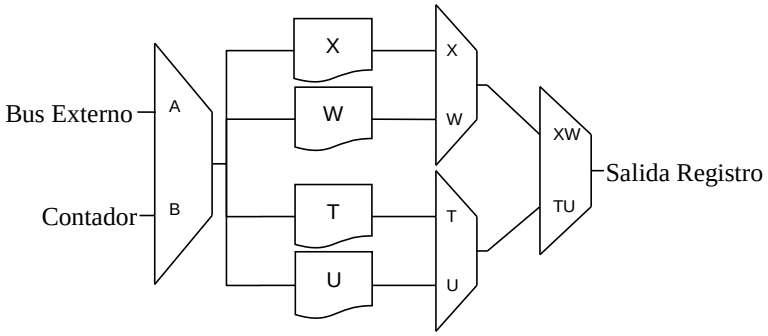


Figura 5.8: Diagrama en bloques de los registros y los multiplexores de lectura.

- Registro X: Este registro se puede habilitar de dos formas distintas, según se desee almacenar datos externos o copiar el valor del contador. La primera opción se logra por medio de la selección de la celda, (a través de las señales correspondientes) y la segunda al seleccionar la salida del contador.
- Registro W: El registro W se selecciona de tres formas diferentes, según se desee almacenar datos externos, copiar el valor del contador, o copiar un dato externo en función del valor del comparador de la celda. Este registro se habilita con la selección de la celda y con la salida del comparador, de manera que si la salida del comparador es "1" el registro se habilita para su escritura (cuando el valor del registro seleccionado es mayor que el valor que se encuentra en el BUS de entrada) y el dato a almacenar puede provenir del exterior o del contador.
- Registro T: El registro T se habilita con la salida del comprador.

Las entradas de datos de los registros X, U y W son seleccionables entre el Bus de datos externo o la salida del contador. La entrada del registro T son los 4 bits menos significativos del Bus externo.

Cada uno de los registros está formado por un arreglo de latches. La Fig. 5.9 muestra la máscara de un latch individual. La Fig. 5.10 muestra la máscara del banco de registros, incluyendo el multiplexor de entrada, que selecciona entre el bus externo y el contador, y los multiplexores de selección de registro. El tamaño de cada latch es de aproximadamente $5\mu m^2$, y el de los multiplexores de $10\mu m^2$, siendo el tamaño total del banco de registros de $185\mu m^2$.

SelCell	CbusB	HabAn	Comparador	Habilitación
1	x	x	x	1
0	1	x	x	1
0	0	1	1	0
0	0	1	0	1
0	0	0	x	0

Tabla 5.1: Habilitación del registro U

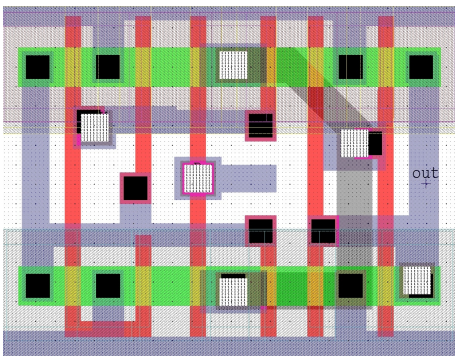


Figura 5.9: Máscara de un latch.

Motor de procesamiento PWL

El motor de procesamiento PWL es el circuito encargado de realizar el procesamiento de la información en la celda. Este bloque se encarga de realizar la comparación del valor del registro seleccionado con la rampa de programa, seleccionar las celdas de la vecindad a las cuales se le aplicarán las funciones F(Vf) y G(Vg), y calcular la operación lógica entre F(Vf) y G(Vg).

El valor del próximo estado es el resultado de la operación lógica entre dos funciones F(Vf) y G(Vg), aplicadas a dos registros. Ambas funciones pueden ser aplicadas a cualquiera de los cuatro registros. Con un multiplexor 4:1 de seis bits, se selecciona a que registro se le aplicará la función F(Vf) y a cual se le aplicará la función G(Vg). En el caso que se seleccione el registro T, los dos bits más significativos son ”0”. Para realizar la comparación del valor del registro con la rampa de programa, la celda cuenta con un solo comparador el cual se multiplexa en tiempo para realizar las

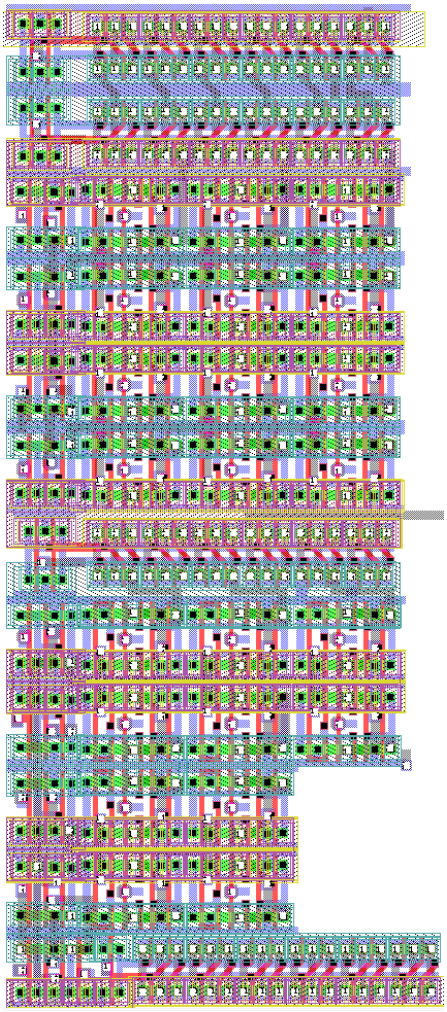


Figura 5.10: Máscara del banco de registros.

comparaciones. El valor de la comparación produce las señales $F_{(pwm)}$ y $G_{(pwm)}$, que se almacenan en dos registros de un bit según se esté comparando el registro correspondiente a la función F(Vf) o G(Vg). El esquemático del comparador se muestra en la Fig. 5.11.

Para realizar la operación lógica entre los valores de F(Vf) y G(Vg), se necesitan dos pasos, primero se selecciona el registro (X, U W o T) al que se le aplicará la función F(Vf), se realiza la comparación y se almacena la señal $F_(pwm)$. Luego se selecciona el segundo registro al que se le aplicará la función G(Vg), se realiza la

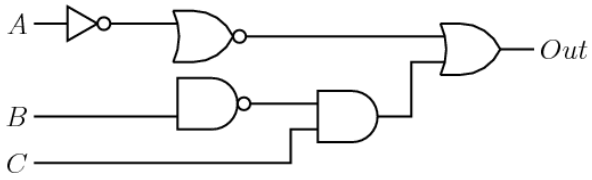


Figura 5.11: Esquemático del comparador digital usado para la generación de la palabra PWM.

comparación y se almacena el valor $G_(pwm)$. Las señales $F_{(pwm)}$ y $G_{(pwm)}$ provenientes de todos los vecinos se concatenan para formar dos vectores llamados Vf y Vg. La selección de la vecindad (es decir, la selección de los vecinos cuyos valores de $F_{(pwm)}$ y $G_{(pwm)}$ van a concatenarse) se hace por medio de dos multiplexores, que seleccionan las diferentes entradas de los vecinos. La salida de estos multiplexores son los vectores Vf y Vg, que apuntan a la dirección de memoria a la cual se debe ir a buscar el valor de la función. Para la función F(Vf) la vecindad se puede seleccionar entre los cuatro vecinos en cruz (+) o equis (×) . Para la función G(Vg) la dirección del vértice puede provenir de los cuatro vecinos en forma de cruz (+) o del valor del registro T.

La memoria donde se almacenan los valores de los vértices puede ser accedido por todas las celdas. Como la configuración de vecindad elegida es de cinco celdas, y el valor de cada vértice es de un bit, la cantidad de señales que van a cada celda son sesenta y cuatro (64), es decir, treinta y dos (32) por cada función. Para que la celda lea el valor de la memoria, se utilizan 2 multiplexores de 32 a 1, cuya entrada a los multiplexores son los vectores Vf y Vg, y la salida es el valor de la función en ese vértice. Para obtener el valor final de F(Vf) operado con G(Vg) , se los opera lógicamente por medio del bloque FoG.

El bloque FoG, es una unidad lógica programable, donde se configura el tipo de función lógica a aplicarle a las dos entradas. Las opciones son AND, OR , XOR o "0". La máscara del bloque FoG se muestra en la Fig. 5.12 y su tamaño es de aproximadamente $5\mu m \times 5\mu m$. La Tabla 5.2 muestra la tabla de verdad del bloque FoG.

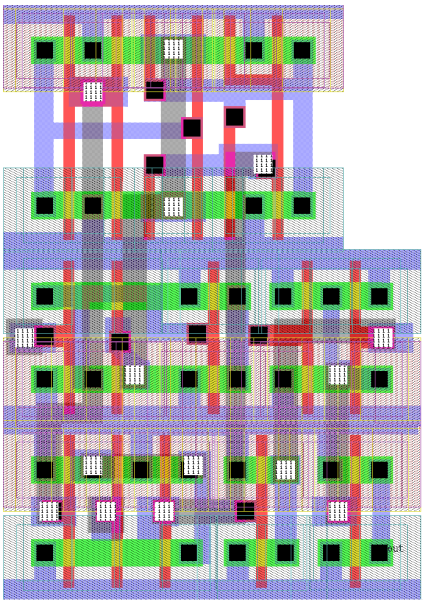


Figura 5.12: Máscara del bloque FoG.

Fog(0)	Fog(1)	Salida
0	0	0
0	1	$A * B$
1	0	$A \oplus B$
1	1	$A + B$

Tabla 5.2: Tabla de verdad del bloque FoG.

Combinando distintas configuraciones de vecindad y de operación entre funciones se pueden lograr muchos casos de procesamiento. La tabla 5.3 muestra todas las posibles combinaciones de funciones y vecindades que se pueden aplicar a las funciones F(Vf) y G(Vg).

La salida del bloque FoG indica el valor que se debe integrar, y para realizar esta tarea se utiliza un contador de seis bits. Luego de finalizada la rampa de procesamiento, (en este caso 63 ciclos por cada registro) en el contador se encuentra almacenado el nuevo valor de estado (el valor de la imagen procesada). Este valor se puede almacenar en los registros X, U o W o para habilitar el registro T. Para

$F(+)+G(+)$	$F(+)+GT$	$F(\times)+G+$	$F(\times)+GT$
$F(+)*G(+)$	$F(+)*GT$	$F(\times)*G+$	$F(\times)*GT$
$F(+)\oplus G(+)$	$F(+)\oplus GT$	$F(\times)\oplus G+$	$F(\times)\oplus GT$

Tabla 5.3: Combinaciones posibles de funciones y vecindades

almacenarlo en el registro W se debe cumplir con la condición de que el valor del contador debe ser menor que el valor del registro seleccionado (registro que este entrando al comparador) dado que la habilitación depende del comparador. La Fig. 5.13 muestra la máscara del contador y del comparador, donde se puede ver que son casi del mismo tamaño.

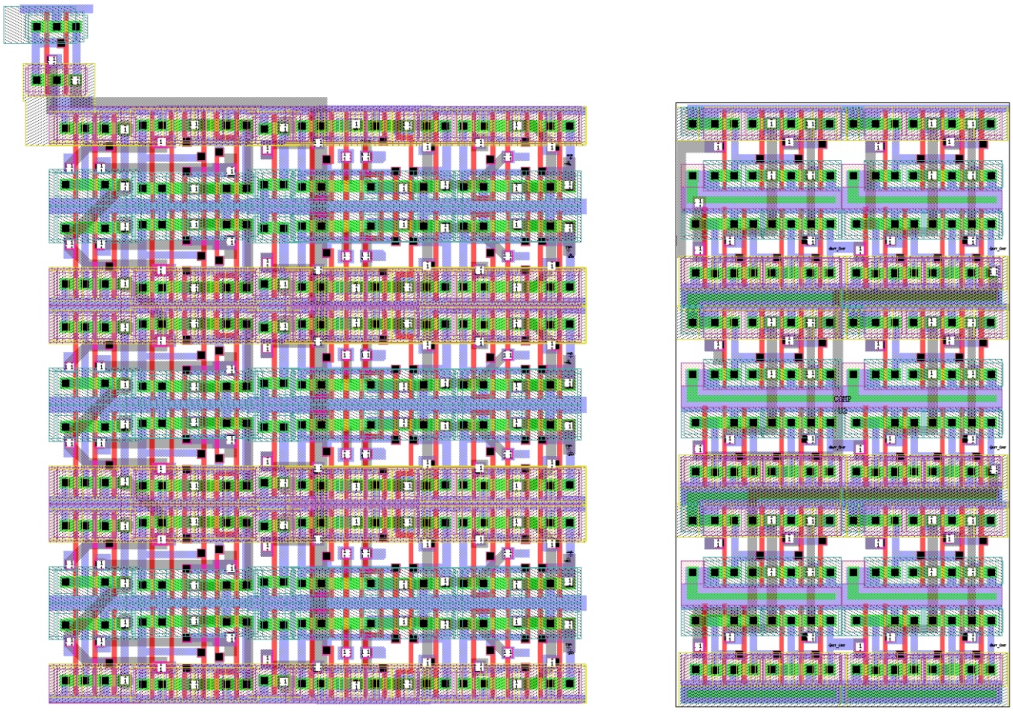


Figura 5.13: a)Máscara del contador b)Máscara del comparador.

El motor de procesamiento PWL ocupa un casi un 50 % de la celda. La máscara completa de este bloque se muestra en la Fig. 5.14. El tamaño del bloque FoG es de $12\mu m^2$, el tamaño del comparador es de $65\mu m^2$, el tamaño del contador es de $114\mu m^2$, el área de los multiplexores es de $61\mu m^2$, el área de los multiplexores que

seleccionan el área de vecindad es de $12\mu m^2$, el área del motor de procesamiento PWL sin el contador es de $250\mu m^2$, y con el contador es de $360\mu m^2$, El tamaño total de la celda es de $25 \times 25 = 625\mu m^2$.

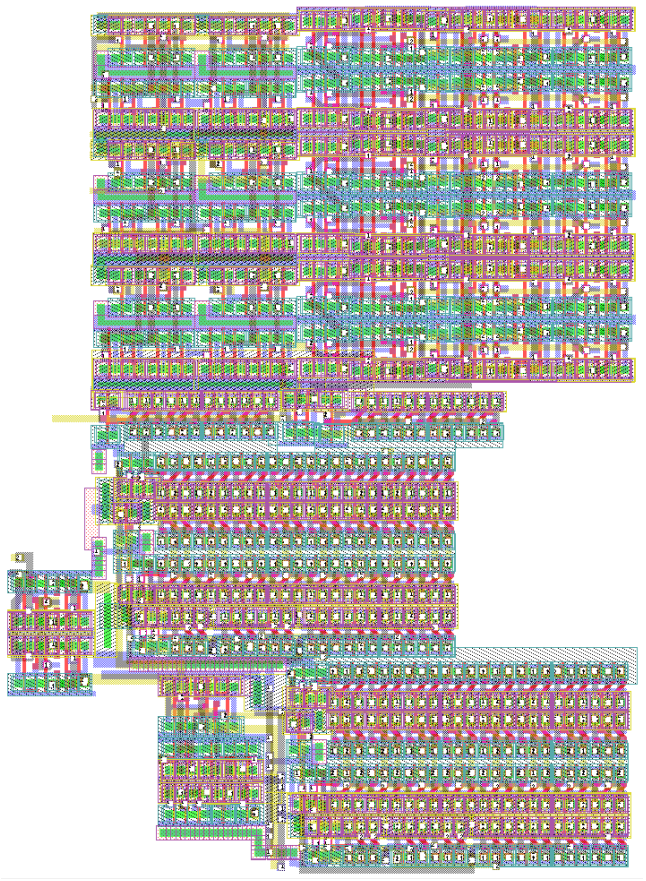


Figura 5.14: Máscara del motor de procesamiento PWL.

Buffer de salida

Cada celda se encuentra conectada a una línea de bus que interconecta las celdas de una misma fila de todas las columnas. La celda seleccionada toma control del bus y escribe el valor del registro seleccionado, cuando se activa la señal de lectura. En otros casos la celda tiene su salida en alta impedancia. Cada celda posee 6 buffers de alta impedancia para escribir en el bus. En el bus se escribe el valor del registro

seleccionado para ingresar al comparador. La máscara de los 6 buffers, se muestran en la Fig. 5.15. El tamaño del buffer de salida es de $33\mu m^2$

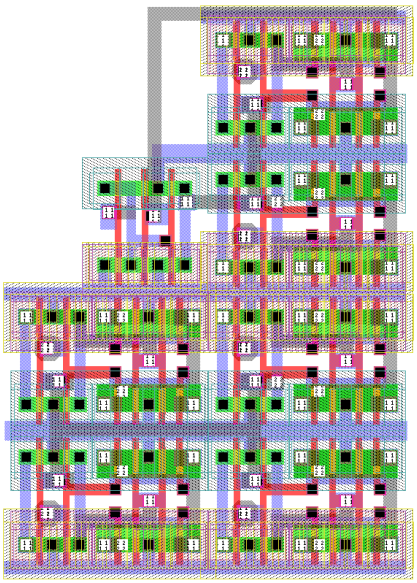


Figura 5.15: Máscara del arreglo de buffers de la celda.

Resumen Celda

Un diagrama en bloque completo de la celda se muestra en la Fig. 5.16, donde se puede ver la interconexión de los bloques. A cada celda ingresan más de 120 señales a través de 5 capas de metal, provenientes del exterior, de la memoria o de las celdas vecinas.

La celda completa tiene un tamaño de 25x25um. Una imagen de la máscara de una celda, se muestra en la Fig. 5.17, donde se pueden ver todas las estructuras explicadas anteriormente.

- Tamaño banco de registros: $185\mu m^2$
- Tamaño Pixel: $40\mu m^2$
- Tamaño Motor de procesamiento PWL: $360\mu m^2$

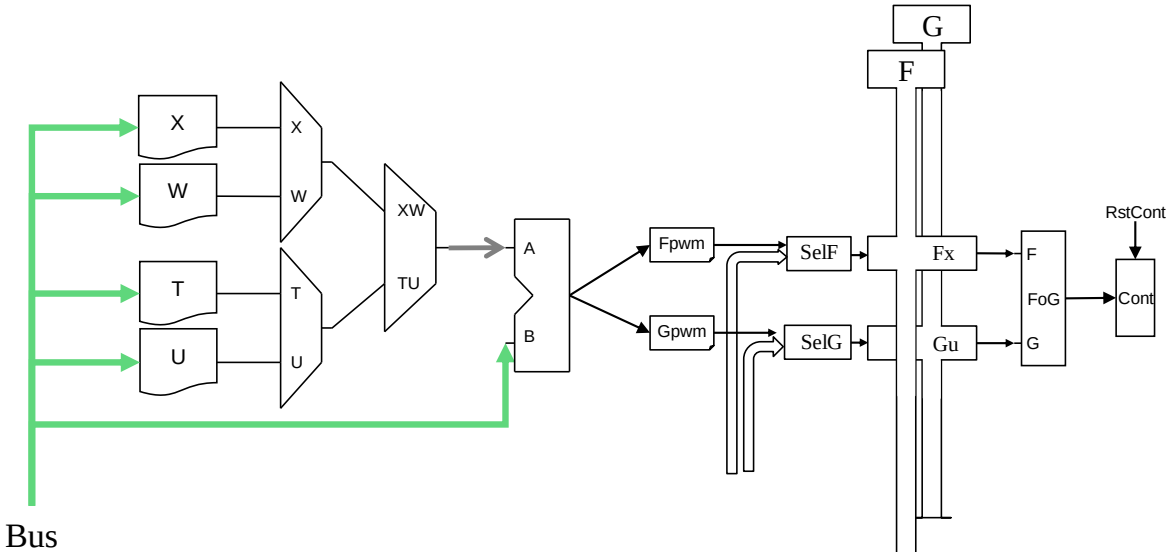


Figura 5.16: Diagrama en bloque de la celda completa.

- Tamaño Buffer de salida: $33\mu m^2$
- Tamaño total de celda: $625\mu m^2$

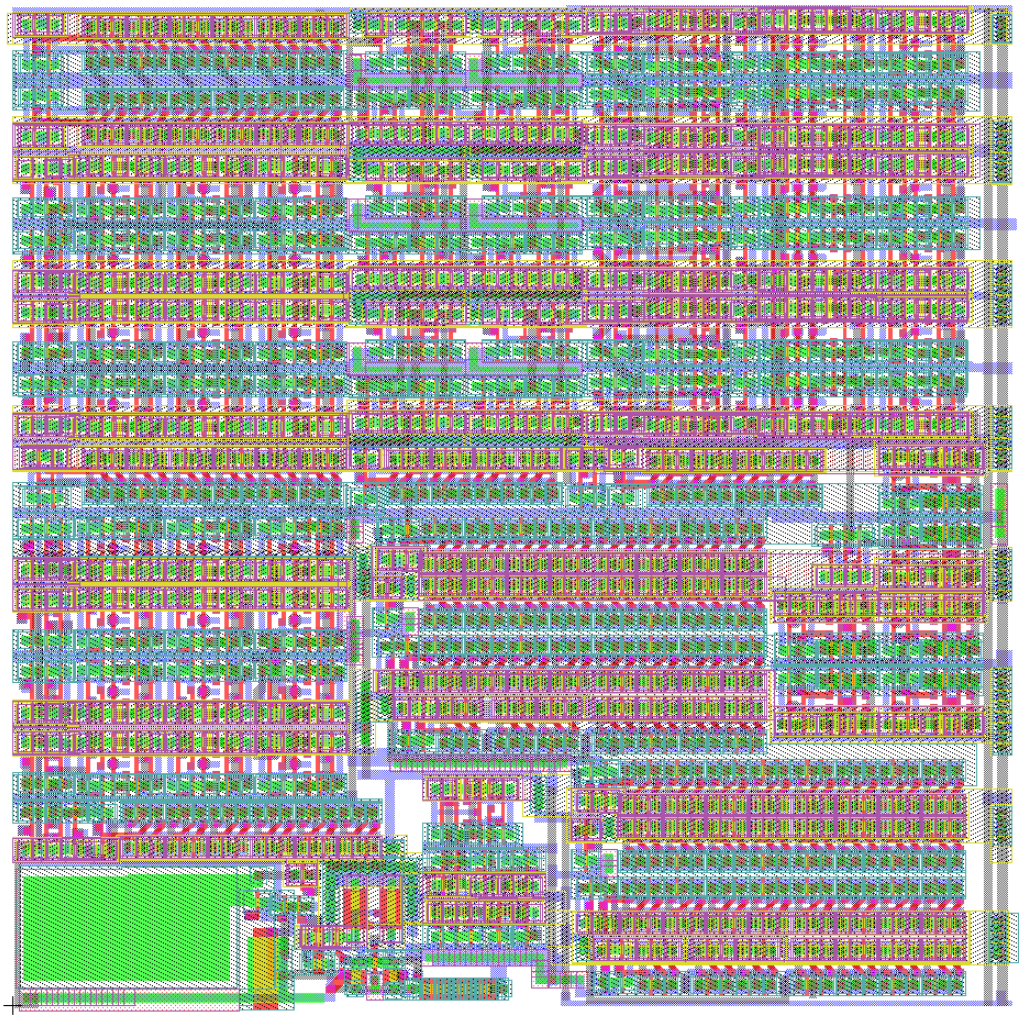


Figura 5.17: Máscara celda completa.

5.3.3. Bloques extras

En esta sección se muestran otros circuitos dentro del chip, que no están dentro del arreglo de celdas. Primero se describe la arquitectura del circuito sumador, que se utiliza para obtener la suma de los valores de las celdas seleccionadas. Luego se describen los pads y los buffers de señales internas.

Sumador de columna

Al seleccionar una columna, todas las filas escriben el valor del registro seleccionado en su correspondiente bus de fila. Hay tantos buses de 6 bits como filas hay en el circuito y cada uno de esos buses se conecta a un sumador de 11 bits. Una de las entradas de los sumadores se conecta al bus de la fila correspondiente y la otra entrada a la salida del sumador de la fila anterior, tal como se muestra en la Fig. 5.18.

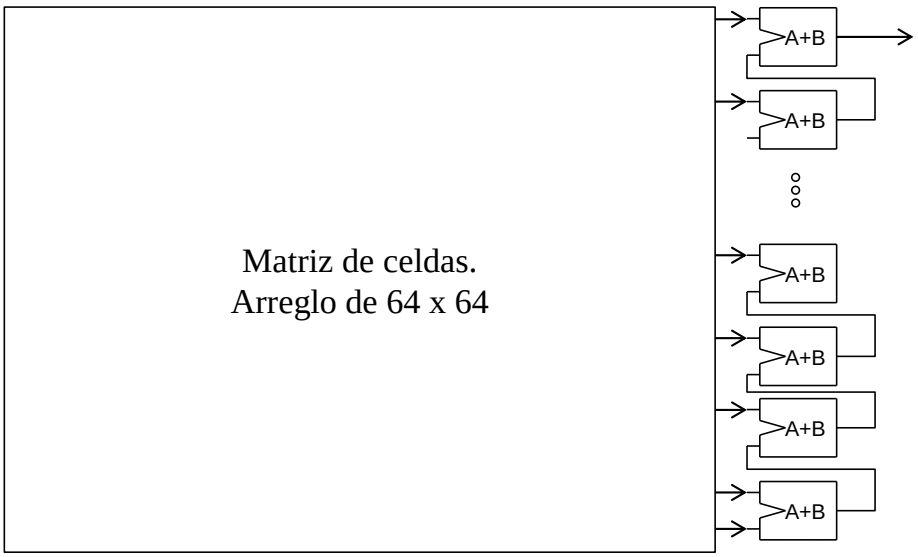


Figura 5.18: Diagrama en bloques del sumador de filas.

Los sumadores son todos iguales y poseen una entrada de 6 bits, donde se conecta el bus de la fila, una entrada de 11 bits, donde se conecta la salida del sumador anterior y una salida de 11 bits. Se diseñaron todos iguales para que la misma estructura se repita fila por fila. Esta estructura se puede realizar de una manera

más eficiente, comenzando con sumadores de 6 bits, y a partir de las filas, 2, 4, 8,16 y 32 ir aumentando un bit en el sumador. Quedando el de la fila 1 de 6 bits, los de la fila 2 y 3 de 7 bits, los de la fila 4 a la 7 de 8 bits, los de la fila 8 a la 15 de 9 bits, los de la fila 16 a la 32 de 10 bits y los de la fila 32 en delante de 11 bits.

Memoria de programa

La memoria, donde se almacenan los valores de la función en los vértices, como se ha dicho anteriormente, es de 32 bits, y se almacena en un registro serie. Hay 2 registros series de 32 bits, uno para cada función. Para enviar la información a todas las celdas se almacena el valor en un arreglo de latches los cuales a través de los buffers correspondientes envían la función a todas las celdas. Esto permite ir cargando una función mientras se ejecuta otra. La máscara de un registro, un latch y un buffer del arreglo de 32 se muestra en la fig 5.19.

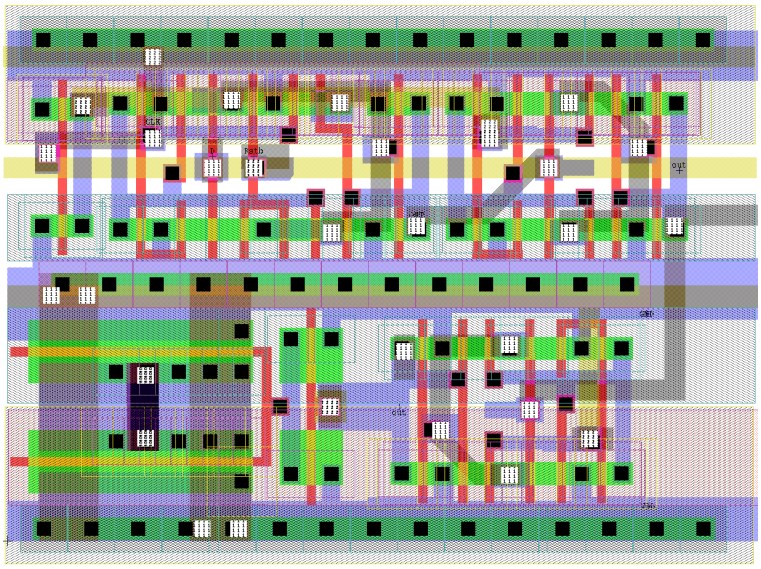


Figura 5.19: Máscara de un bloque sumador.

5.3.4. Pads

Los pads son los dispositivos que permiten al chip comunicarse con el exterior. Estos poseen varias estructuras como un gran cuadrado de metal llamado "*Bond*", donde se conecta el cable al encapsulado, y estructuras para proteger contra descargas electrostáticas. En este chip, hay cuatro tipos distintos de pads, de alimentación, de entrada, de salida y auxiliares. Los pads de alimentación son diferentes dependiendo si alimentan al anillo de pads o a la lógica dentro del chip, también llamada "*core*". Los pads están diseñados para trabajar entre 1V y 2,4V. La alimentación del core del chip puede variar entre 1v y 1,2v. La alimentación del chip está separada en 4 zonas, el core, dos grupos disconexos para alimentar a los pads de entrada, y una para alimentar a los pads de salida. Los Pads de Entrada y de Salida poseen buffers, para poder manejar las capacidades dentro y fuera del chip. También cuentan con circuitos especiales para proteger el core de descargas electroestáticas. Los Pads Auxiliares cuentan con protección electroestática y están conectados (directamente cableados) a la salida de los pads de entrada. Estos pads, permiten verificar que efectivamente los pads de entrada escriban correctamente. En caso de un mal funcionamiento, los pads de entrada se pueden deshabilitar, (por zonas con la alimentación) y así poder ingresar señales directamente al chip por medio de los pads auxiliares.

La Fig. 5.20 muestra los layouts de los pads, (a) es un pad de alimentación, (b) es un pad de entrada con un auxiliar, y (c) es un pad de salida.

Buffer de señales internas

Muchas de las señales externas deben ingresar directamente a todas las celdas del arreglo. Teniendo en cuenta que el arreglo es grande y se debe cumplir con las restricciones de tiempo, las señales se distribuyen a las celdas a través de un arreglo de buffers. Las señales se distribuyen por filas, y un arreglo de buffers distribuyen las señales a todas las columnas, habiendo en cada fila tantos buffers como señales a distribuir.

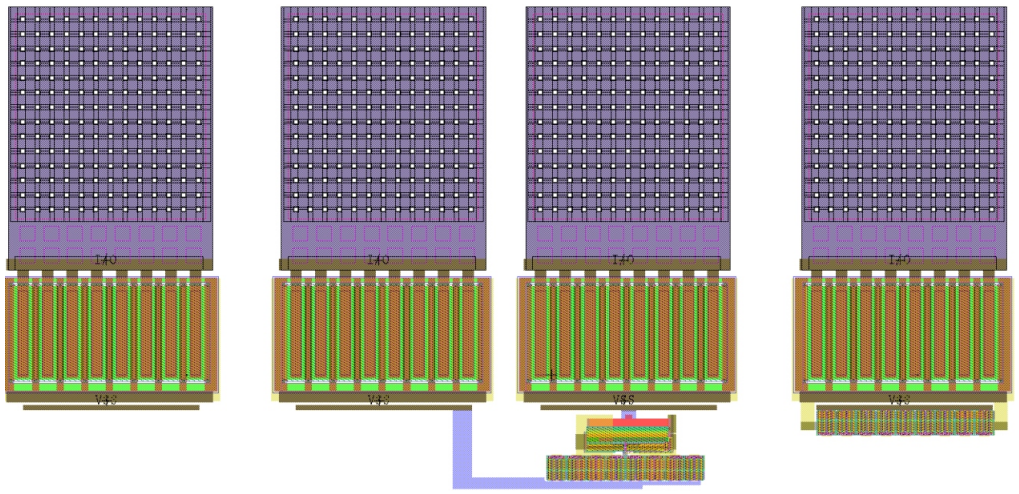


Figura 5.20: Máscara de los pads que permiten la comunicación con el exterior del chip.

5.4. Floorplaning

En esta sección se describe la ubicación e interconexión de cada uno de los bloques dentro del chip. Esta tecnología tiene 2 capas de metal de sección gruesa en los niveles superiores y 7 capas de metal de sección más delgada por debajo. Las capas de mayor sección poseen menor resistencia, se utilizan para distribuir alimentación a todas las celdas y para conectar las señales globales que se distribuyen a todo el arreglo. Los pads están distribuidos alrededor del chip, formando un anillo. La alimentación se distribuye a todo el chip a través de los niveles de metal 8 y 9. Las señales de entrada al circuito se distribuyen partiendo desde la izquierda del arreglo. En cada fila se encuentra un grupo de buffers, que envían las señales a todo el arreglo por medio de metal 7. En la parte superior del arreglo, se encuentran los 2 registros series, los latches, que almacenan las funciones F y G, y los buffers, que la distribuyen a todo el arreglo por medio de metal 6. La salida del circuito se encuentra a la derecha del arreglo. Al seleccionar una columna todas las celdas de la columna escriben en su fila correspondiente, en una línea de metal 7, y fuera del arreglo, se encuentran los buffers que se habilitan con la señal de fila, que escriben en un bus vertical de metal

5. La interconexión entre los vecinos se hace por medio de metal 5 y 4. Los metales 4, 3, 2 y 1 se utilizan para interconexiones dentro de la celda. Los metales pares interconectan de manera vertical, y los metales impares, tienen una distribución horizontal. La Fig. 5.21 muestra la máscara del chip completa.

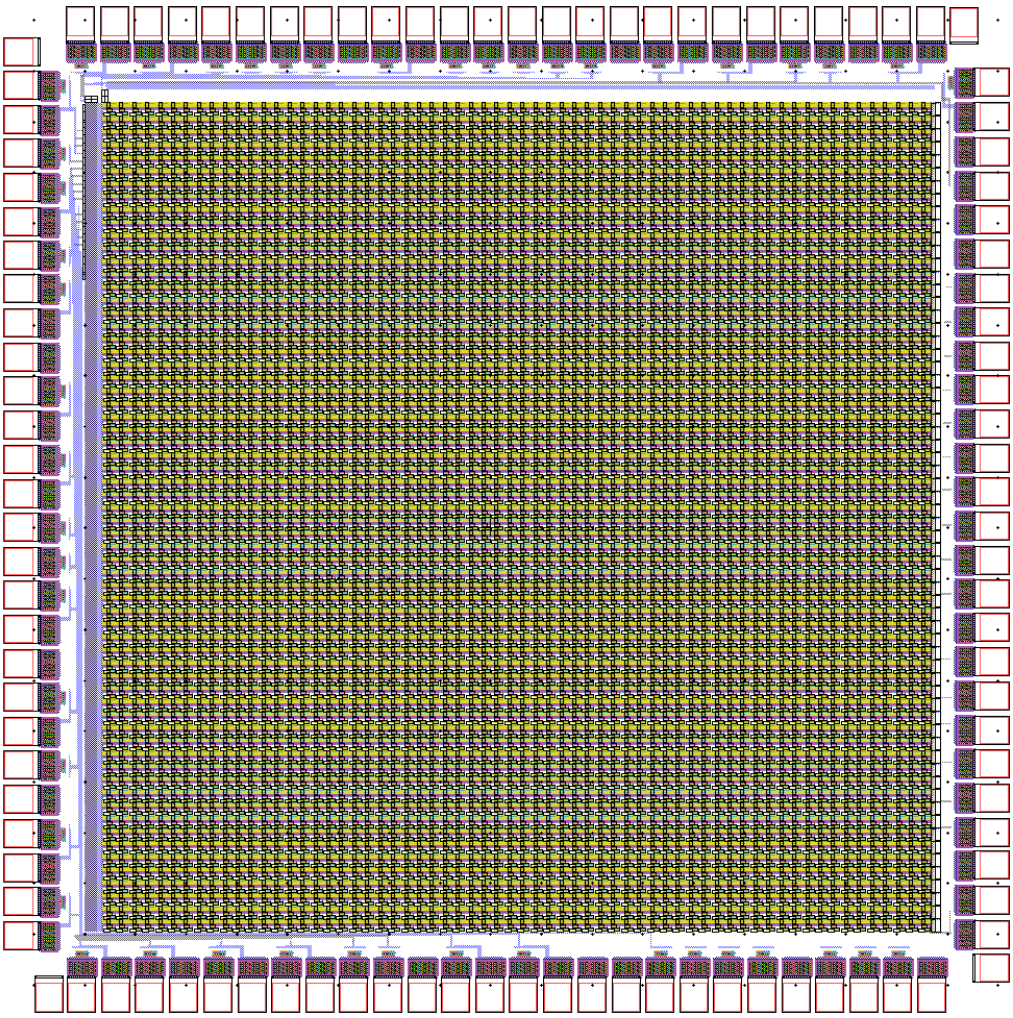


Figura 5.21: Máscara del circuito completo.

5.5. Funcionamiento

En esta sección se describen las señales que controlan el circuito, se explican sus funciones y se ilustra paso a paso como realizar algunas de las tareas básicas del

circuito. También se muestran resultados de simulaciones, testeos realizados previos a la fabricación y datos experimentales obtenidos del chip ya fabricado.

5.5.1. Señales

El imager posee entrada y salida de Bus, para escribir y leer datos de las celdas, señales de configuración y control para habilitar distintos bloques o partes del circuito y señales de reloj que hacen funcionar distintos bloques. A continuación se enumeran todas las entradas y salidas que tiene el circuito, y se explica brevemente la función que cumplen.

Señales de Datos

- BusIn: Este bus de 6 bits se utiliza para guardar valores en los registros, para la rampa de programa y para la realización de la conversión AD.
- Fin, Gin: Entradas de 1 bit, para cargar las Funciones F y G en los registros serie.
- BusOut: Salida de 6 bits del registro de la celda seleccionada.
- SumOut: Salida de 11 bits del sumador.
- AnalogIn: Entrada Analógica para realizar la Conversión AD utilizando el conversor de rampa simple.

Señales de configuración

- SelVecF y SelVecG: Seleccionan el tipo de vecindad que se utilizará para aplicar la función F y G.La Tabla 5.4 muestra el tipo de selección de los vecinos en función de la entrada seleccionada.
- SelWTXUb, SelXWUb: Seleccionan los registros que se van a operar o a leer. La Tabla 5.5 muestra la selección de los registros U, X, T y W en función de las entradas.

Señal	Valor	Tipo Selección
SelVecF	0	$+$
	1	X
SelVecG	0	$+$
	1	T

Tabla 5.4: Selección de vecinos

SelWTXUb	SelXWTUb	Registro
0	0	U
0	1	X
1	0	T
1	1	W

Tabla 5.5: Habilitación de los registros

- FoGin: Selecciona la operación lógica que se aplicará entre los valores de F y G. La Tabla 5.6 muestra la operación lógica que se aplicara a los valores de la funciones de F y G en funcion de la entrada $FoGin$.
- Bordes: Valor de la vecindad de las celdas que se encuentran en los bordes.
- VDDPixel: Tensión de inicialización a la cual se carga el diodo.

FoGin	Operación
00	AND
01	OR
10	XOR
11	0

Tabla 5.6: Función Lógica a aplicar en función de la señal FoGin.

Señales de control

- Row, Col: Señales de direccionamiento (BUS de de 6 bits) de la celda de fila y columna respectivamente.
- SelCBusb: Selecciona si las entradas a los registros proviene del bus de entrada o del valor del contador de cada celda.
- HabComp: Habilita el registro U y el comparador, para almacenar el dato de la conversión Analógica.
- Read: Habilita los buffers de escritura de las celdas.

Señales de reloj

- LatchX, LatchW, LatchU, LatchT: Señales para almacenar valores en los registros seleccionados.
- LatchF, LatchG: Almacena el valor del comparador en el registro F o G para generar el vértice del simplice donde buscar el valor de la función.
- ClkC: Clock de los contadores de las celdas.
- Fclk, Gclk: Entradas para cargar las Funciones F y G en los registros serie.
- ClkCmp: Clock del comparador del conversor AD
- SHPixel: Muestrea la tensión del fotodiodo en todos los pixeles.
- RstPxl: Inicializa los fotodiodos.
- RstCnt: Resetea los contadores de todas las celdas.
- Dscsmpl: Descarga el capacitor donde se almacena el valor de la muestra de tensión del píxel.

5.5.2. Tareas Paso a Paso

Las tareas que el circuito puede realizar son: almacenar un valor en un registro, leer el valor de un registro, capturar una imagen, realizar la conversión AD de la imagen capturada, cargar una función en la memoria, procesar una imagen y sumar el valor de todos los registros de una imagen completa. A continuación se describen brevemente como realizar algunas de estas tareas.

Procedimiento de Escritura de una imagen o de un valor en un registro

Como primer paso se debe seleccionar la celda destino con las entradas Col y Row. Una vez hecho esto, se habilita el registro correspondiente con las señales correspondientes. Se coloca el dato a almacenar en la entrada $BusIn$, y se aplica un pulso al reloj del registro correspondiente. Para almacenar una imagen completa en el arreglo, se debe direccionar una por una las celdas e ir almacenando el valor correspondiente. Si se desea almacenar el valor del contador en algún registro, se coloca la señal $CBusB$ en "1", se habilita la celda en la que se quiere almacenar y se aplica un pulso de reloj. En este caso, todas las celdas del arreglo copian el valor del contador en el registro habilitado. La Fig. 5.22 muestra el procedimiento para copiar los datos del contador al registro W, y como almacenar datos en los registros X, U y T. Se debe notar que solo se almacenaran en el registro W, los datos del contador que sean menores que los almacenados previamente.

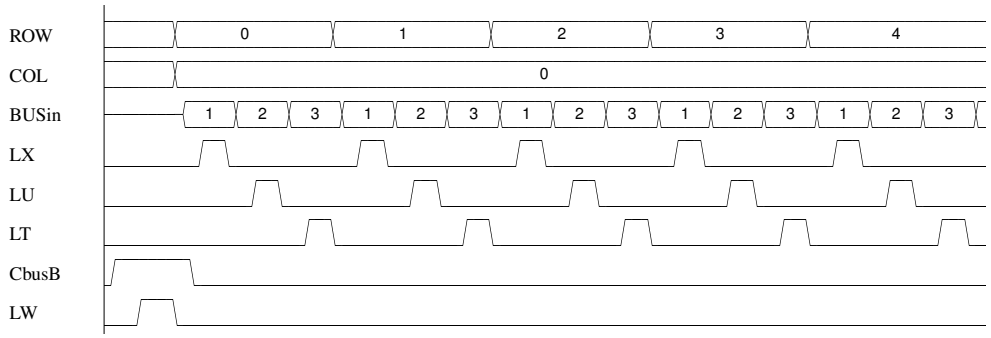


Figura 5.22: Secuencia se señales para almacenar datos en los registros.

Procedimiento de Lectura de una imagen o del valor de un registro de una celda

Como primer paso se debe seleccionar la celda con las entradas Col y Row. Se selecciona también por medio de las señales $SelWTXUb$, $SelXWTUb$ el registro a leer. Una vez hecho esto, se habilita el buffer de salida con la señal $Read$, el valor del registro de la celda seleccionada se encuentra en la señal $BusOut$. Para leer una imagen completa en el arreglo, se debe direccionar una por una las celdas e ir leyendo el valor correspondiente. Si se desea leer el valor del contador, primero se debe almacenar en un registro y luego leer el registro.

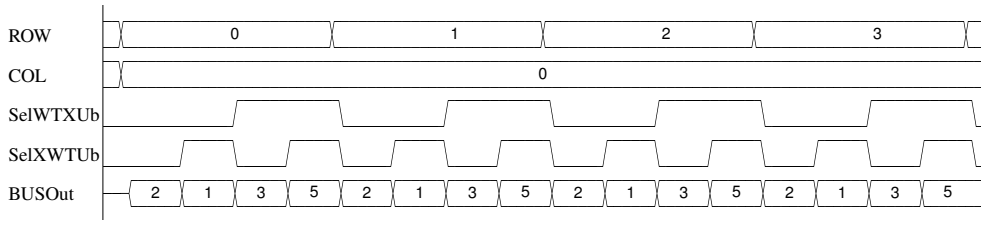


Figura 5.23: Secuencia se señales para leer datos de los registros.

Procedimiento para capturar una imagen y realizar su conversión a digital

Al comienzo del ciclo de adquisición se carga el fotodiodo a una tensión de inicialización fijada desde el exterior. Para esto se coloca la tensión deseada en la señal $VDDPixel$ y se aplica un pulso en la señal $RstPxl$. Luego se coloca el Bus digital, $BusIn$, en cero y la tensión analógica, $AnalogIn$, en la tensión correspondiente en la cual se realice la comparación. En un siguiente paso se abre el transistor que resetea al pixel, dejando al diodo integrar fotones libremente. Se habilita el comparador a través de la señal $HabComp$ y se equilibra el comparador colocando la señal $ClkCmp$ en "1". Al colocar la señal $ClkCmp$ en "0" la salida del comparador toma un valor digital "0" o "1" en función de la comparación de la rampa analógica con la entrada $AnalogIn$. Luego del tiempo de establecimiento del comparador (este tiempo no supera los *nanosegundos*), se aplica un pulso a la señal LU, para almacenar el valor en el registro U. Luego se incrementa el valor de la rampa digital, se aplica un pulso

a la señal *ClkCmp*, y se aplica un pulso a la señal *LU*, el ciclo se repite para todos los valores de la rampa digital desde 0 hasta 62. Luego se deshabilita el comparador colocando en "0" la señal *HabComp* y en el registro U queda almacenado el valor de la conversión AD. La Fig. 5.24 muestra la secuencia de señales para realizar la conversión AD.

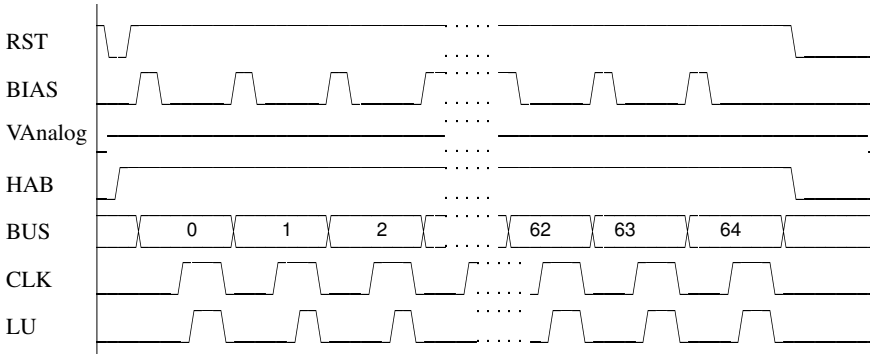


Figura 5.24: Secuencia se señales para realizar la conversión AD.

Procedimiento para cargar las Funciones Función F(Vf) y G(Vg)

Las funciones que evaluará el sistema se almacenan en latches y la forma de ingresarlos es a través de un registro serie. Para ingresar datos, se coloca el valor de la función F (G) en la entrada Fin (Gin), del vértice 00000 y se aplica un pulso de reloj a la señal *Fclk* (*Gclk*). Luego se coloca el valor de la función F (G) en la entrada *Fin* (*Gin*), del vértice 00001 y se aplica un pulso de reloj a la señal *Fclk* (*Gclk*). Esto se repite hasta el vértice 11111. Una vez cargado el registro serie con los valores de la función se latchean para enviarlos a todas las celdas con la señal *LatchFG*. La Fig. 5.25 muestra la secuencia de señales para cargar una función F.

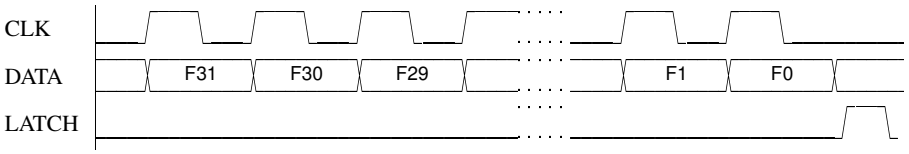


Figura 5.25: Secuencia se señales para cargar una función en memoria.

Procedimiento de Procesamiento

Para realizar el procesamiento de la imagen, como primer paso se debe cargar la función que se desea aplicar (tanto F(Vf) como G(Vg)). Se inicializa el contador de las celdas, a través de la señal $RstCnt$. Luego se selecciona el tipo de vecindad a la cual se le va a aplicar la función con las señales $SelVecF$ y $SelVecG$, se establece el valor de las entradas de los vecinos de los borde a través de la señal $Borde$, y se selecciona el tipo de operación que se le aplicara a las funciones F(Vf) y G(Vg), a través de la señal $FoGin$. Luego de haber inicializado las celdas y de establecer las señales de configuración del procesamiento se comienza con la rampa de programa. Se coloca el bus ($BusIn$) en cero, se selecciona el registro al cual se le aplicara la función F(Vf) , con las señales $SelWTXUb$, $SelXWTUb$ y se latchea el valor de la comparación con la señal $LatchF$. Luego se selecciona el registro al cual se le aplicara la función G(Vg) , con las señales $SelWTXUb$, $SelXWTUb$, y se latchea el valor de la comparación con la señal $LatchG$. Una vez realizado esto, los multiplexores que leen la memoria tendrán los valores de las funciones de los vértices correspondientes y la salida del bloque FoG ya tendrá el valor de la función que se aplicará. El siguiente paso es realizar la integración del valor de la función aplicando un pulso en la señal $ClkC$. Luego se incrementa el valor del Bus, y se repite el procedimiento hasta que el valor del Bus llegue a 62. Al finalizar la rampa de procesamiento, el valor del nuevo estado queda almacenado en el contador. Como último paso se debe cargar el valor del contador en un registro. La Fig. 5.26 muestra la secuencia de señales para realizar el procesamiento.

Procedimiento para obtener la suma total de valores en imagen

Este procedimiento se utiliza para realizar la suma de todos los valores de los registros de todo el arreglo. Si la imagen esta binarizada, valiendo el fondo 0 y el objeto 1, este procedimiento sirve para calcular el área del objeto y si se aplica un reconocimiento de bordes, y se realiza este procedimiento, la salida es el perímetro del objeto medido en cantidad de pixeles. Estos valores son importantes para obtener descriptores de imagen, tal como se explicó en el capítulo 2. Para comenzar, se debe aclarar que se necesita un acumulador externo para obtener el resultado final de la

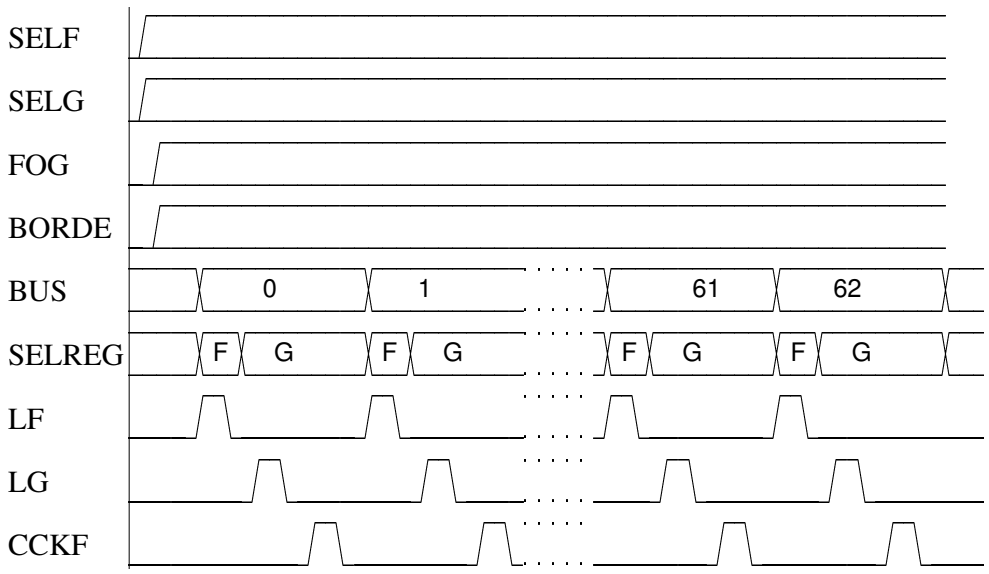


Figura 5.26: Secuencia se señales para realizar el procesamiento de una imagen.

suma y se debe inicializar en cero. Luego se selecciona la primera columna y la salida del sumador se almacena en el acumulador externo. Luego se selecciona la siguiente columna y se suma el valor del sumador al valor del acumulador. Se realiza este procedimiento hasta que se recorren todas las columnas. Al recorrer toda la imagen (seleccionar la última columna) el valor del acumulador contiene el valor de la suma de todos los valores de las celdas del arreglo.

5.5.3. Verificación.

Para realizar las verificaciones de su funcionamiento se han realizado simulaciones en con el software de desarrollo Cadence. Las simulaciones se realizaron utilizando el motor de simulación UltraSim. Como primer paso se simularon las celdas básicas y luego el arreglo completo. La Fig. 5.27 muestra la simulación del funcionamiento del bloque FoG. La señal de salida es la operación lógica seleccionada entre las entradas SelfFog(0) y SelFoG(1).

La Fig. 5.28 muestra el proceso de escritura y lectura de un dato en el registro X. Se ve que si se aplica la señal de latch sin estar habilitado el dato no se guarda, y al habilitarse el registro el dato queda guardado.

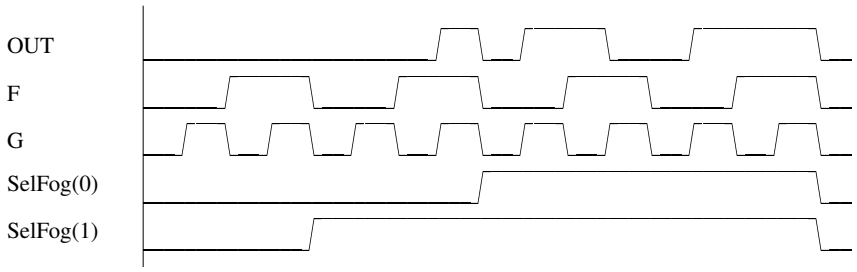


Figura 5.27: Simulación del bloque FoG.

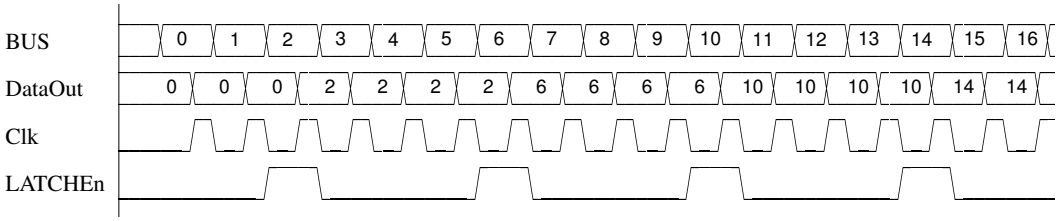


Figura 5.28: Simulación del proceso de escritura y lectura de un registro.

La Fig. 5.29 muestra la señal salida del comparador (señal Fpwm) al realizar la comparación entre el registro X, y la entrada del bus. Se ve que la salida es "1" hasta que el valor del bus alcanza el valor almacenado en el registro X, y luego cambia a "0".

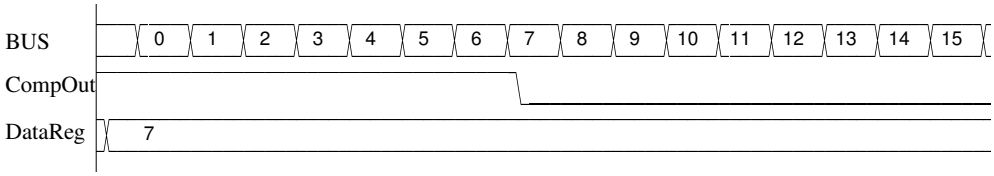
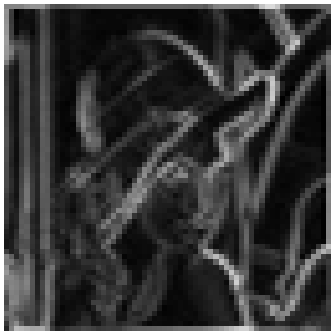


Figura 5.29: Simulación del bloque comparador.

La información del testeo de este circuito se encuentra en el Cap.7. Se cargaron imágenes desde el exterior y se procesaron con varias funciones. La Fig. 5.30 muestra un ejemplo de procesamiento donde se calcula los bordes de una imagen.

a)



b)

Figura 5.30: a) Imagen original; b)Imagen procesada.

5.6. Conclusión

En este capítulo se ha mostrado la realización de un imager, con capacidad de procesamiento S-CNN fabricado en una tecnología de $90nm$. Se ha explicado paso a paso el proceso de diseño y la arquitectura del circuito junto con algunos de los circuitos y sus máscaras de fabricación. También se ha incluido una descripción del plano de planta o "*floorplanning*", indicando la ubicación de cada uno de los bloques circuitales. Se ha explicado el funcionamiento de la estructura, detallando la actividad que cada uno de los bloques realiza en cada uno de los pasos necesarios para realizar varias tareas. Por último, en este capítulo se han reportado mediciones experimentales del circuito integrado. Las mediciones convalidan los resultados teóricos esperados, y evidencian el correcto funcionamiento del chip. La tabla 5.7 muestra un resumen de las características del Imager. Se debe trabajar mucho más en el diodo y en la etapa de conversión, dado el FPN es demasiado amplio. Se puede analizar la opción de crear un solo conversor AD en vez de uno por pixel.

Imager	
Tecnología	CMOS $90nm$ 7 metales 1 Poly
Arreglo	64×64
Tamaño del integrado	$2mm \times 2mm$
Imágenes	Escala de Grises
Precisión	$\sim$ 6 bits
Memorias de imagen	4
Vecindad	de 4 (Seleccionable entre + y x) y registro T
Número de transistores	5,3 10^6
Tamaño celda	$25\mu \times 25\mu m$
Fill Factor	5 %
Estructuras Extras	Sumador de valores de Columna
Velocidad de trabajo	150Mhz
Mínima cantidad de clocks para procesar	1
Máxima cantidad de clocks para procesar	128

Tabla 5.7: Características de la cámara inteligente

Capítulo 6

Realización 3D

6.1. Introducción

En este capítulo se describe el desarrollo de un sensor de imágenes con capacidad de procesamiento S-CNN fabricado en una tecnología 3D sobre un proceso de $130nm$. Las tecnologías 3D interconectan varias obleas de silicio en forma vertical, utilizando conexiones metálicas entre las obleas, o vías, cuyos diámetros son del orden de las centenas de micrómetros. De esta manera, la tecnología 3D permite incrementar la densidad de integración al permitir construir un chip múltiple con diferentes pisos, denominados "tiers". La tecnología de integración 3D es una alternativa novedosa para la creación de Sistemas en Chip (SoC) [27], por las varias ventajas que posee [34, 33], entre las que se pueden destacar una mayor densidad de integración lineal con la cantidad de tiers (mayor cantidad de transistores por unidad de volumen, y la posibilidad de interconectar obleas de diferentes tipos de tecnología como las DRAM, CMOS o CCD [43, 23, 26]. Por otro lado, la posibilidad de ubicar bloques circuitales en tres dimensiones, produce una reducción en las interconexiones, lo cual traer aparejado una disminución de las capacidades parásitas y un incremento asociado de la velocidad de operación.

Para el diseño de este Imager, se planteó realizar un sistema que realice tareas primarias, tareas de alto nivel y que incluya comunicación con otros módulos de manera de poder contar con la posibilidad de que varios de estos circuitos trabajen paralelamente.

El Imager diseñado consta de un arreglo rectangular de 48×32 celdas (cuya región de vecindad es seleccionable en forma de cruz (+) o equis ($\times$)), dos memorias de programa, un sumador de columna, una unidad de multiplicación y división, un generador de códigos cadena, un correlador de cadenas de 16 valores y un microprocesador 8051. Cada una de las celdas del arreglo posee dos registros, un sensor activo y un comparador para realizar la conversión A/D de la señal de luz, un par de registros para almacenar la señal PWM y el integrador. Gran parte del motor de procesamiento PWL es compartido por todas las celdas de la misma fila. Este sistema cuenta además con un sistema de interconexión externa que permite que varios circuitos integrados operen en forma coordinada, para la ejecución de tareas complejas. Las especificaciones del diseño se desarrollaron con el objetivo de minimizar el tamaño de la celda y aprovechar al máximo las características que brinda el proceso de integración 3D. Para obtener una implementación eficiente en términos del área ocupada por cada celda, se colocó una única memoria en la periferia del chip.

El capítulo está organizado como se detalla a continuación. En la Sección 6.2 se describe paso a paso el flujo de diseño seguido y la utilización de las herramientas para el desarrollo, síntesis, simulación, creación de máscaras y verificación de diseño. En la Sección 6.3 se explica en detalle el tipo de arquitectura seleccionada, las celdas del arreglo y las estructuras extras que brindan soporte al arreglo. En esa sección también se detalla la distribución de cada uno de los bloques del circuito y su interconexión. En la Sección 6.5 se enumeran las señales que controlan el circuito, se explican sus funciones y se ilustran paso a paso algunas de las tareas básicas que realiza el circuito. Se muestran también simulaciones y ensayos realizados previos a la fabricación. En la Sección 6.6 se encuentran las conclusiones y un resumen con las características más sobresalientes del circuito.

6.2. Diseño

Para realizar el diseño de este circuito se siguió un flujo de diseño conocido como "Top-Down", donde se comienza por una descripción de alto nivel del circuito y utilizando software de diseño automatizado se crea una máscara donde se colocan

e interconectan todas las compuertas lógicas creando un "mar de compuertas"que tiene el comportamiento de la descripción lógica realizada. La secuencia de pasos seguidos en el diseño se muestra en la Fig. 6.1. En primer lugar, se seleccionó la arquitectura de la celda, realizando el análisis de las posibles configuraciones tal como fuera explicado en el Cap. 4. Dado que el chip se debe distribuir en dos pisos, se dividió el diseño en dos entidades diferentes, una por cada "*piso*" o "*Tier*", distribuyendo los bloques para equilibrar la cantidad de lógica en cada uno de los pisos. En segundo lugar se describió el comportamiento de la celda y de las estructuras de soporte en lenguaje VHDL. Se creó un arreglo de 8×8 celdas, el cual se sintetizó con el software de desarrollo ISE® de Xilinx, y se implementó en una FPGA Spartan 3E para testear su correcto funcionamiento y capacidades de procesamiento. Para realizar esta verificación, se utilizó una interfaz programada en MATLAB®, la cual se comunica a través del puerto serie de la computadora con un microprocesador PicoBlaze® sintetizado en la FPGA. En un siguiente paso, se modificó el VHDL para optimizarlo para la creación de máscaras, dado que hay variaciones dependiendo si el destino del circuito descripto será una máscara o una implementación sobre una FPGA. Se sintetizaron ambos "*tiers*" con el software Design Vision®, de Sinopsys®, utilizando compuertas estándares de la tecnología de $130nm$ de la empresa Chartered. Luego se realizó la colocación y conexionado (Place and Route) de las compuertas, utilizando el paquete de software Encounter®. Para la ubicación y alineación correcta de cada una de las máscaras se utilizó el paquete Virtuoso® de Cadence®. Las simulaciones de todos los bloques sintetizados, incluyendo la interconexión entre los dos pisos, se realizó con el software ModelSim®.

6.3. Arquitectura

El chip fue fabricado en una tecnología de $130nm$ de la compañía Chartered y la interconexión entre las dos obleas fue realizada por la compañía Tezzaron utilizando una técnica de flip chip, donde las dos obleas se interconectan una con la otra a través de la última capa de metal. El sistema completo consiste de un arreglo de 48×32 celdas donde cada celda posee dos registros, un contador, un elemento fotosensor y un conversor A/D. La región de vecindad de cada celda es de cinco

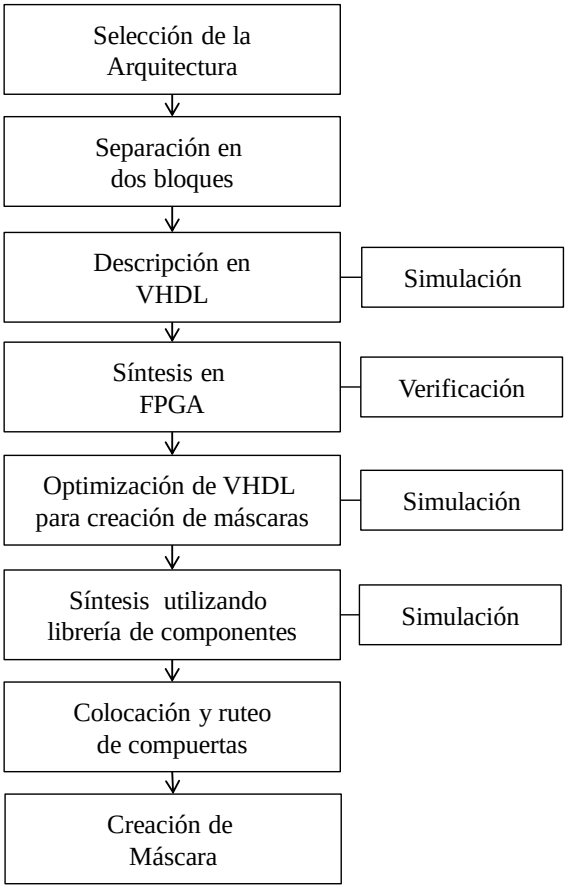


Figura 6.1: Diagrama de flujo del diseño seguido.

vecinos, seleccionable en forma de cruz (+) o equis (×). Los valores de los estados de las celdas de los bordes son configurables por lados, pudiendo colocar un lado en "1"mientras los demás están en "0". Las funciones que se pueden aplicar son dos, F y G, cuyo valor es de un bit y se almacenan en dos registros serie de 32 bits. El valor de estas dos funciones puede ser operado con las funciones lógicas producto AND, suma OR o suma exclusiva XOR. El comparador para generar la señal PWM y los multiplexores que obtienen el valor de la función en el vértice son compartidos por todas las celdas de la misma fila. Al compartir el motor de procesamiento, la memoria no debe ser distribuida a todas las celdas, sino que se debe distribuir a todos los bloques de procesamiento que se encuentran fuera del arreglo.

Se han colocado varias estructuras adicionales que realizan cálculos matemáticos

y ejecutan algoritmos primarios como el cálculo del código cadena de la imagen, el cálculo de la correlación entre dos cadenas de dieciséis valores de 16 bits y un microprocesador 8051, que posibilita controlar el circuito completo. Se cuenta también con un puerto de comunicaciones RS232 para enviar señales hacia y desde el exterior. También hay dos puertos de comunicación, para interconectar varios circuitos integrados idénticos trabajando simultáneamente. La Fig. 6.2 muestra un diagrama con todos los bloques del sistema.

El chip completo abarca un área de $2mm \times 2,5mm$, donde se han ubicado en total 70 pads, con un voltaje de trabajo de 3,3V. En este diseño se utilizaron compuertas estándar para la parte digital, y se realizó un diseñó a medida para el APS y el comparador del conversor A/D. La arquitectura se eligió para que sea simple y de poca dificultad de ruteo, ya que para su ubicación y conexionado se utilizaría un software de diseño.

6.3.1. Selección de la arquitectura

La selección de la arquitectura tuvo como finalidad optimizar el tamaño de la celda, y dotar al circuito con la capacidad de realizar tareas de alto nivel. El arreglo de celdas solo realiza procesamiento de bajo y medio nivel. Para procesar imágenes a más alto nivel es necesario el uso de estructuras extras. Debido a esto, se minimizó el tamaño de cada celda de manera de poder contar con un arreglo denso de celdas, a la vez de contar con un espacio separado para la realización de tareas de alto nivel. A estos efectos, se maximizó la utilización de recursos compartidos de manera de lograr una celda compacta. La selección de la vecindad, la forma de distribuir la memoria a las celdas, la cantidad y la precisión de los registros fueron seleccionadas tal como se describe a continuación:

Selección de la vecindad: Se separó la vecindad en dos grupos de cinco vecinos, quedando la posibilidad de conectar los cuatro vecinos a la celda central en forma de + o en forma de $\times$, tal como fuera explicado en el Cap. 4. De esta forma se logra que cada celda tenga conectividad con sus nueve vecinos, y se disminuye el tamaño de la memoria de $2^9 = 512$ a $2^5 = 32$.

Registros: Cada celda posee 2 registros de 7 bits. La selección de los 7 bits se realizo para poder almacenar la posicion de cada pixel dentro del arreglo. Cuando se ejecuta un algoritmo cuyo valor final depende de la posición de la celda en el arreglo, cada celda debe poder almacenar en uno de sus registros el valor de su posición, ya sea fila o columna. Dado que el arreglo es de 48×32 son necesarios siete bits para poder almacenar estos valores. Para realizar la interconexión entre "*tiers*", las conexiones deben estar ubicadas en lugares específicos definidos por el fabricante, las cuales forman una grilla hexagonal con una distancia entre conexiones de 5. Con siete bits, el tamaño final de la celda es de 25×25. Al hacer los registros de 8 bits, el tamaño de la celda debe aumentar a 30×30 (un paso más en la grilla) quedando la celda más grande y con varios espacios inutilizados, debido a que el aumento de la celda es mayor que el de la lógica. Esto significa que para un incremento del doble de presición (1bit)corresponde un 44 % de aumento de área, lo cual produce una pérdida considerable de área, además de un desperdicio. Por esto, los registros se diseñaron de siete bits.

Recursos compartidos: El tiempo en procesar una imagen depende de la precisión de los registros. En este caso, como los registros son de siete bits, una imagen se procesa como máximo en 128 pasos. Si el circuito opera con un reloj a una frecuencia de 150Mhz, puede procesar más de un millón de imágenes por segundo. Para tareas de bajo nivel este número es elevado [19], por esto se decidió compartir el motor de procesamiento PWL con todas las celdas de la misma fila. Esto produjo una reducción del tamaño de las celdas al de casi un 50 %, y el tiempo de procesamiento se redujo 48 veces (se multiplexa en tiempo el acceso al motor de procesamiento) dado que para cada paso de la *Rampa de programa* todas las celdas de la fila (en este caso 48) deben acceder al motor de procesamiento.

Distribución de la memoria: Dado que los motores de procesamiento PWL se encuentran fuera del arreglo, un incremento en su tamaño no tiene impacto en las dimensiones de la celda ni en el tamaño del arreglo. Por este motivo se envían todas las líneas de memoria a los motores de procesamiento. Como se explicará en la Sección 6.5 la distribución de la memoria, y la precisión de los registros hacen que el

circuito procese como máximo en 12098 ciclos de reloj y como mínimo en 50 ciclos de reloj dependiendo la función a realizar.

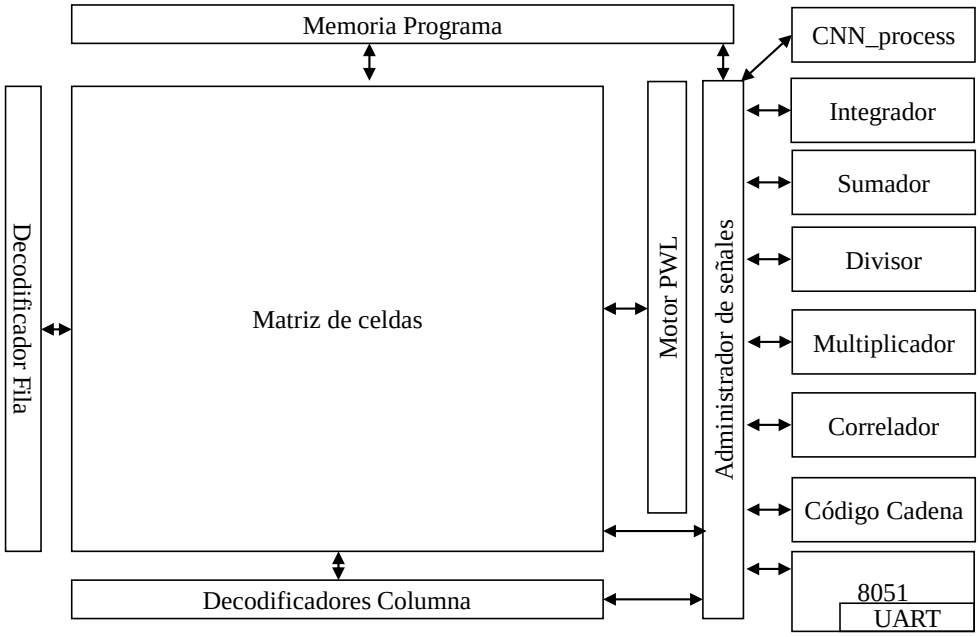


Figura 6.2: Diagrama de bloques del chip completo.

Las características principales de la arquitectura se pueden resumir de la siguiente manera:

- Vecindad:
  Consta de 5 celdas, seleccionable entre tipo cruz (+) o equis (×).

- Cantidad y precisión de registros:
  Cada celda posee dos registros de siete bits

- Recursos compartidos:
  El motor de procesamiento PWL se encuentra parcialmente compartido por todas las celdas de la misma fila, este bloque consta del comparador para generar la señal pwm y el bloque de lectura de memoria del valor de la función.

- Recursos propios de la celda:
  Cada celda tiene sus propios registros, estructuras de habilitación e integrador.

- Estructuras Extras:
  Como estructuras extra se han colocado un sumador por cada fila, un integrador, un multiplicador, un divisor, un generador de código cadena, un correlador de cadenas de 16 bits y un microprocesador.

- Condiciones de Borde:
  Pueden fijarse en "0" ó "1" para los distintos bordes de la imagen. Mientras un borde tiene valor "0" otro puede tener valor "1".

- Función transición:
  Las funciones que se pueden aplicar a los registros son dos, llamadas F y G, cuyo valor es de un bit y el valor de estas dos funciones puede ser operado con las funciones lógicas AND, OR o XOR.

- Memoria:
  Dado que la vecindad está definida por 5 celdas, la memoria debe ser de $2^5 = 32$ valores. En este caso cada uno de los valores de la memoria es de un bit, siendo cada una de las memorias de programa de 32 bits. La memoria está conformada por dos registros serie de 32 bits, ubicados fuera del arreglo y todos sus valores se distribuyen a los motores de procesamiento compartidos por las celdas.

6.3.2. Bloques de la celda

La celda consta de 3 bloques básicos: el bloque de adquisición de imágenes, el banco de registros, y parte del motor de procesamiento PWL. La Fig. 6.3 muestra un diagrama en bloques de cada una de las celdas.

Sensor y conversor A/D

Para la creación del elemento conversor se utiliza un sensor activo (APS) realizado con una isla de implante $n+$ sobre el sustrato tipo p. Este circuito fue diseñado

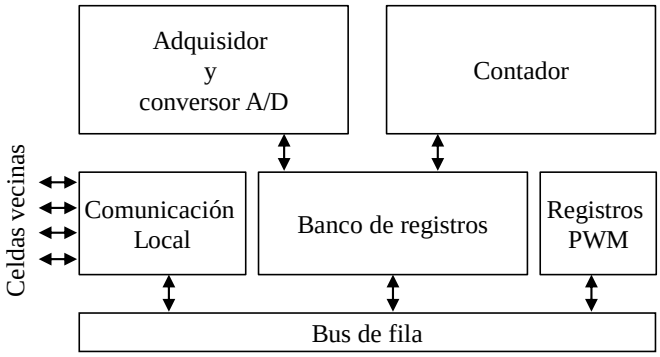


Figura 6.3: Diagrama en bloques de las celdas.

a medida y se insertó en el flujo de diseño automatizado. La Fig. 6.4 muestra la máscara de este circuito.

Para realizar la conversión A/D se utiliza el método de rampa simple, debido a su tamaño reducido, dado que solo requiere de un comparador. El conversor analógico digital utilizado en esta celda consta de un comparador que compara la tensión en el diodo, con un valor de tensión externo en sincronía con una rampa digital. Los valores de tensión en los terminales del diodo disminuyen con una velocidad proporcional a la cantidad de luz incidente. Así, los diodos mas iluminados se descargan más rápido que los menos iluminados. Al tener una tensión externa fija, la medición de luz se puede realizar mediante la medición del tiempo de descarga del diodo.

Al inicio de la conversión el comparador habilita el almacenamiento de la información de la rampa en el registro destino (en este caso, U). De esta manera, en el registro queda almacenado el último valor de la rampa digital, que coincide con el valor digital de la entrada. El comparador utilizado es del tipo conmutado. La comparación se realiza a través de la señal CLK una vez que las señales de entrada se encuentran debidamente establecidas y la salida del comparador ha sido equilibrada. La Fig. 6.5 muestra el esquemático y la máscara del comparador.

Banco de registros

Este bloque digital, descripto en VHDL, consta de dos registros de siete bits, donde cada uno está conectado a través de un buffer de tres estados a una línea

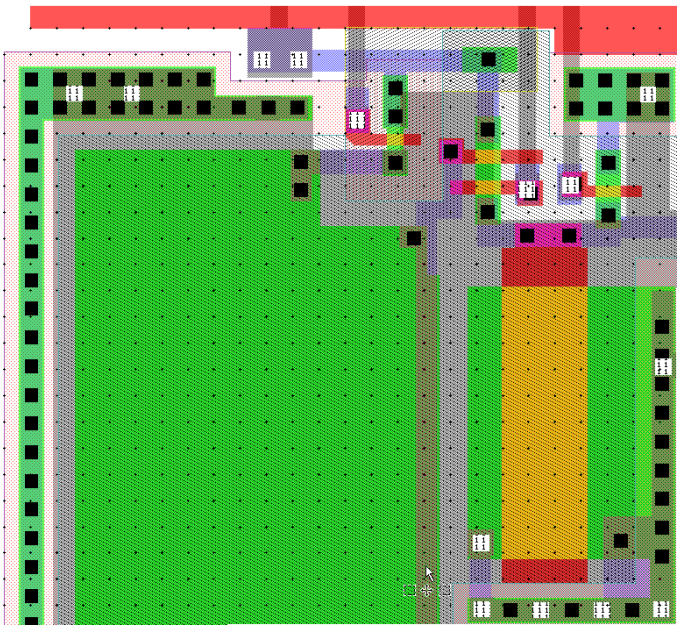


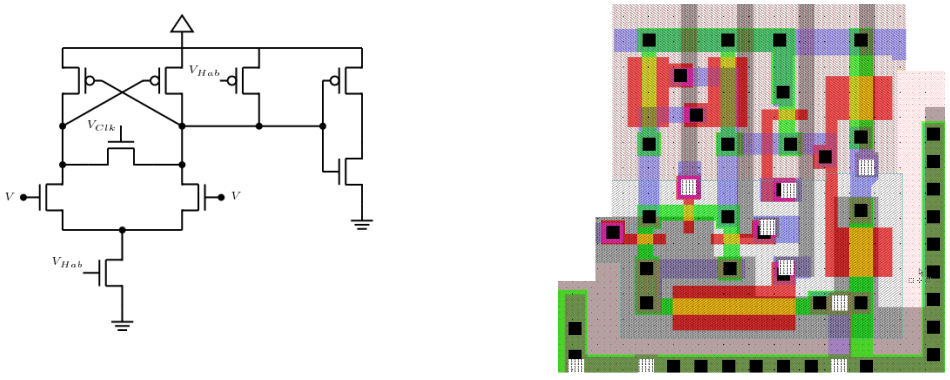
Figura 6.4: Máscara del sensor de luz

de bus compartida por toda la fila. Al habilitar la celda, ésta toma control del bus, enviando la información de los registros al comparador que se encuentra fuera del arreglo y también a los sumadores de fila.

Motor de procesamiento PWL

Parte del motor de procesamiento es compartido por todas las celdas de la misma fila. Los comparadores que generan la señal PWM y los multiplexores usados para leer la memoria se encuentran fuera del arreglo y cada celda solo posee un contador y dos registros para almacenar los valores de F y G.

Los bloques del motor de procesamiento PWL compartidos entre todas las celdas de la fila son los encargados de generar la señal PWM y los que obtienen el valor de la función de la memoria. La Fig. 6.7 muestra la conexión de las celdas de la fila con los bloques compartidos. El Generador PWM evalúa el valor de la señal PWM (comparador) de las filas seleccionadas por la señal HPWM y la memoria guarda las funciones F y G; este bloque está formado por dos registros serie de 32 bits. El bloque de lectura de memoria obtiene el valor PWM de las columnas seleccionadas



a) b)

Figura 6.5: a) Esquemático del comparador usado para la conversión AD 1; b) Máscara del comparador

por la señal HCNT y por medio de dos multiplexores lee la memoria y el bloque FoG. La salida de este bloque está conectada a la entrada de los contadores de su fila.

Al seleccionar una columna todas las celdas de esa columna se conectan a su respectiva estructura enviando el valor de las direcciones de memoria para las dos funciones F y G. El multiplexor selecciona de la memoria el valor correspondiente y lo escribe en la entrada de los contadores. Estas conexiones se pueden ver en la Fig. 6.7 Al recibir un pulso de reloj, los contadores habilitados (columna seleccionada por la entrada), realizan el incremento de su valor, teniendo en cuenta los datos que el multiplexor obtuvo de la memoria operado a través del bloque FoG.

6.3.3. Bloques extras

Este sistema cuenta, con varias estructuras que sirven para aumentar la capacidad de procesamiento y realizan una gran cantidad de tareas específicas. La Tabla 6.1 muestra un resumen de los bloques integrados al sistema y se detallan sus características principales.

Nombre VHDL	Función
MULT	Multiplicador
DIV	Divisor
ADDER	Sumador de valores de Fila
INTEGRATOR	Integrador
CORR	Correlador
CODCAD	Codificador Codigo cadena
MICRO8051	Microprocesador
UART	Sistema de comunicaciones Asincronico
SIGNALMANAGER	Arbitro de señales
CNNPROCESS	Sistema generador de señales para el cálculo automatizado de la función PWL en el arreglo

Tabla 6.1: Lista de bloques extras y sus funciones.

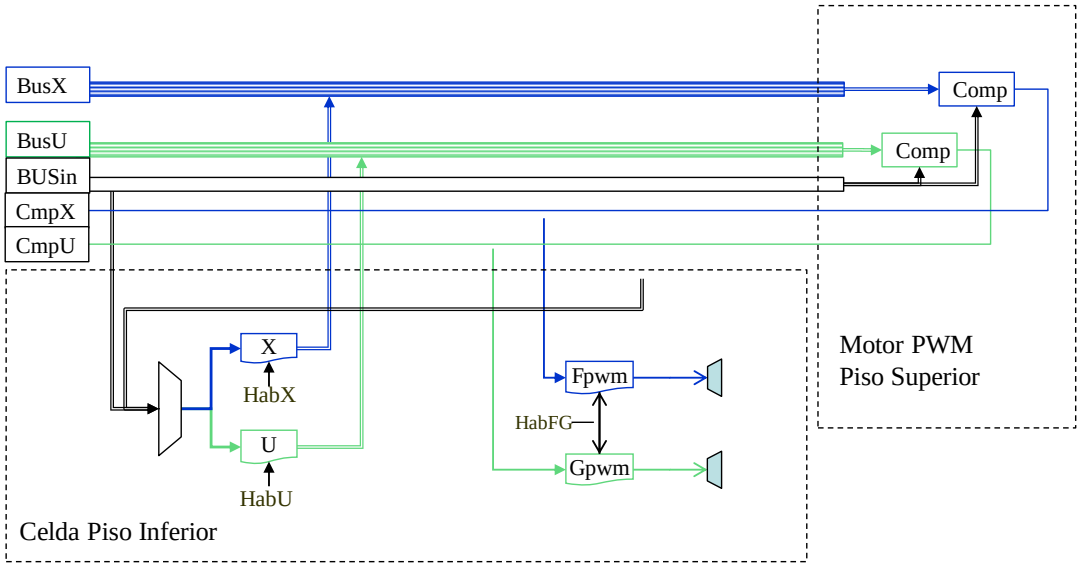


Figura 6.6: Conexión de los registros de las celdas con el comparador.

Multiplicador y Divisor

El multiplicador tiene dos entradas de 8 bits, y una salida de 16 bits. Las estradas están conectadas al Bus Interno, y la salida está conectada al manejador de Bus. El divisor tiene dos entradas de 8 bits y una salida de 16 bits. El vector de salida está separado en parte entera y parte decimal, siendo los 8 bits más significativos la parte entera del resultado, y los 8 menos significativos la parte fraccionaria.

Sumador e Integrador

El sumador tiene como entrada la salida de los valores de todas las filas seleccionadas por la señal HCOL. Hay un bloque sumador por cada fila, donde la entrada de cada uno es la salida del sumador de la fila anterior y el valor del registro de la columna seleccionada, de manera que la salida del último es la suma del valor de los registros seleccionados.

$$Salida = \sum_{fila=1}^{32} Registro \tag{6.1}$$

Este valor es de 12 bits, los cuales se conectan al Bus Interno en dos partes, los

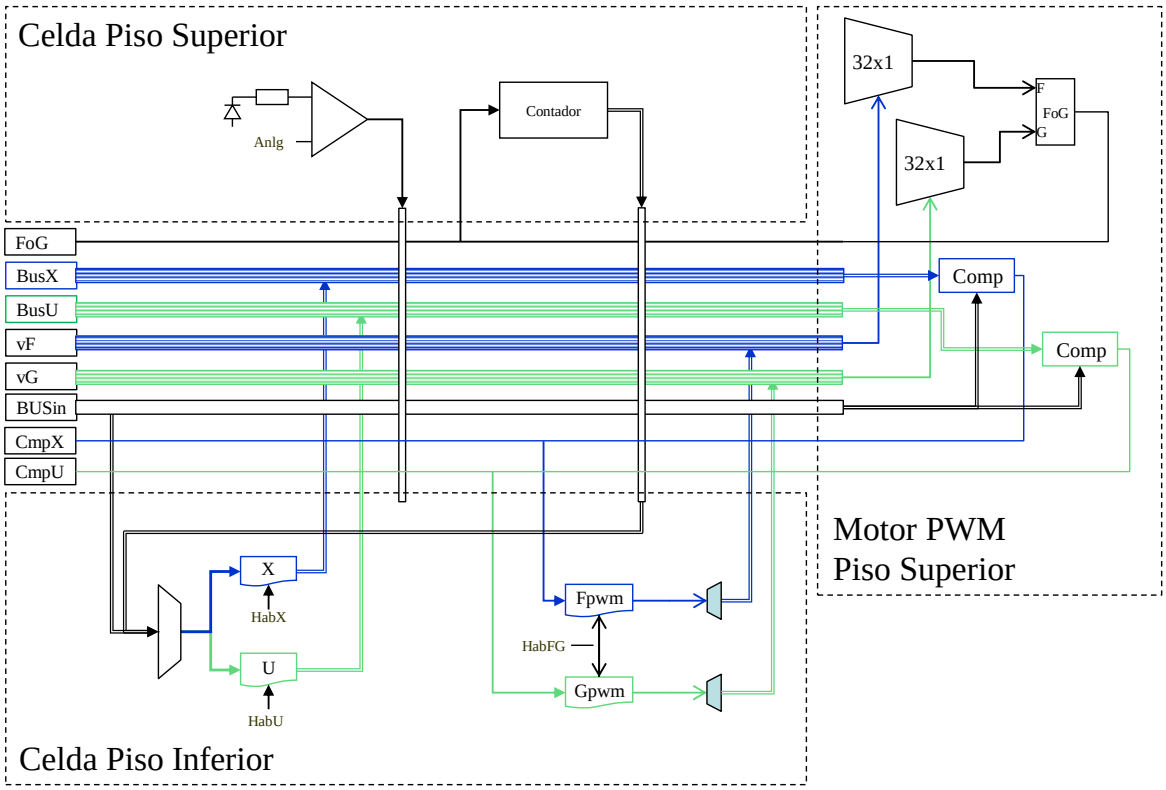


Figura 6.7: Conexión del vector pwm con el bloque lector de direcciones y el contador.

ocho bits menos significativos y los cuatro más significativos.

El integrador está formado por un sumador con un acumulador de 16 bits. La entrada de este bloque está conectada al bus interno. Esta estructura se utiliza para sumar todos los valores de los registros seleccionados en el arreglo. La cantidad de bits del integrador se definió para poder ejecutar el algoritmo de posicionamiento de objetos, para el cual es necesario darle a cada celda el valor de su ubicación, ya sea por fila o columna. Para realizar el algoritmo de posicionamiento, a cada celda se le da el valor de la posición (puede ser fila o columna) se suman todos los valores, y luego se divide el área. Dado que el arreglo es de 48×32 suponiendo que una figura ocupe toda la imagen, el máximo valor máximo será 48x32x49/2=37632, para lo cual se necesitan 12 bits para lograr este número.

Correlador

Este bloque calcula un valor de correlación calculando la suma de diferencias absolutas (SAD) de dos cadenas de 16 valores. El correlador tiene dos entradas, llamadas T y V, y una salida. Las entradas son dos registros serie compuestos por 16 registros de 8 bits cada uno. La salida es la suma de las diferencias entre los valores de los registros T y V. La Fig. 6.8 muestra un diagrama en bloques de este circuito.

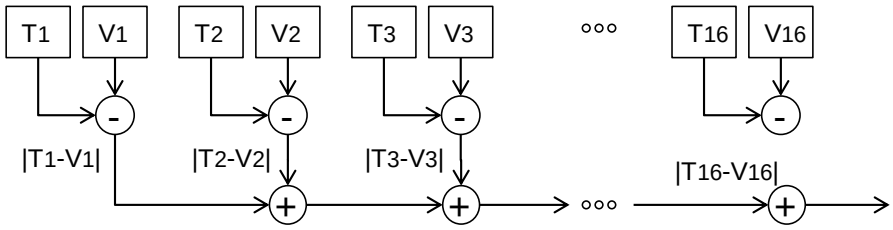


Figura 6.8: Diagrama en bloques del correlador.

$$Salida = \sum_{i=1}^{12} \mid T_i - V_i \mid \tag{6.2}$$

Código Cadena

Este bloque se encarga de realizar el código cadena de la imagen partiendo de una ubicación, y siguiendo una dirección especifica. El bloque cuenta con una unidad de decisión, un generador de direcciones, una unidad de control, un registro serie, y un acumulador. Para seleccionar el valor del próximo píxel a visitar se utiliza la información que viene del píxel indicando cuales de sus vecinos están activos y la dirección actual del movimiento. Si se llega a una bifurcación, la dirección preferencial de movimiento, es la actual, siempre la prioridad es avanzar aumentando filas o aumentando columnas. Para realizar este proceso el bloque necesita la dirección del píxel donde se comenzará (si se desea que sea el comienzo de una línea, se puede utilizar la función "Fin de línea") y la dirección de comienzo de recorrido (Abajo, Arriba, Izquierda o Derecha). Este bloque tiene tres entradas y varias salidas. Una

entrada indica la dirección de fila actual, otra la dirección de columna actual y otra la entrada de datos. Las salidas indican la dirección (fila y columna) del próximo pixel (para ser direccionado y leido), la dirección donde se movió y si hubo una bifurcación. El circuito, al pasar por un píxel con una bifurcación, envía una señal indicando que tuvo que tomar una decisión. El Resultado final del proceso es un registro indicando la cantidad total de píxeles recorridos y los movimientos realizados se almacenan en un registro serie.

Microprocesador y unidad UART

El circuito posee un microprocesador 8051 con un puerto de comunicaciones RS232 y un par de puertos paralelos, para poder interconectar varios Imagers que trabajen en paralelo. Este micro tiene acceso a todas las entradas y salidas de todas las estructuras del sistema, así como de algunos nodos intermedios para realizar una depuración del circuito. El microprocesador cuenta con una pequeña memoria interna de 64 bytes y el programa se debe almacenar en una memoria externa. La dirección de las instrucciones (Instruction Pointer) está conectada al exterior por medio de 12 pines, logrando así, direccionar hasta 4 k bytes de programa. Las conexiones con el circuito se realizan por medio de ocho puertos de ocho bits. Estos puertos son los primeros 8 Bytes de la memoria interna del micro. La Tabla 6.2 muestra las conexiones de cada uno de los puertos en el circuito.

Este bloque permite que el Imager pueda ejecutar todas las tareas necesarias para captar una imagen, procesarla, trabajar en paralelo con otros Imager, y entregar información al exterior, agregando solo una memoria externa donde se almacena el programa.

Bloques de control

Decodificadores: Cada una de las celdas se puede direccionar para ser leída, para que su valor pwm sea evaluado, y para qué el motor de procesamiento PWL, obtenga el valor de la memoria y su contador integre. Cada una de estas tareas se realiza con tres señales distintas, llamadas, HCOL, HPWM y HCNT. Cada una de estas señales tiene su propio decodificador para seleccionar la columa que corresponde. También

hay un decodificador ara seleccionar la fila, en caso que se quiera direccionar una celda especifica del arreglo.

Bloque de procesamiento automático: Para automatizar el proceso de evaluación de la señal pwm y de integración de los valores de la función, se generó un bloque que realiza estas tareas. Este bloque cuenta con 3 entradas, Process PWM, CNT y Comienzo. Las salidas del bloque se conectan a todas las habilitaciones del arreglo, y a los decoders.

Administrador de señales: Hay varias formas de controlar el imager: con señales del exterior, con el microprocesador interno, o con la maquina automatizada. Para seleccionar quien tiene el control de las señales y de los buses, se encuentra el árbitro de señales.

Selección de entradas: La comunicación con el arreglo, se realiza por medio de puertos de 8 bits. Esa comunicación puede provenir desde el exterior o desde el micro, Este bloque habilita la entrada correspondiente de información. Se maneja a través de la señal Microextb

6.4. Floorplaning

En esta sección se describe la ubicación de cada uno de los bloques en el circuito integrado, especificando en que piso se encuentran, y su interconexión. Todas las señales del exterior llegan al piso superior, y se envían al piso inferior a través de conexiones intertier. Dado que esta tecnología es del tipo flip chip, los dos pisos se interconectan entre sí por medio del metal superior (Top Metal). Los bond de los pads, que son las islas de metal donde se conectan los cables del encapsulado, se encuentran en la parte trasera del piso superior, y se conectan al circuito a través de TSV. El piso superior se encuentra expuesto a la luz, por lo tanto allí se colocan los fotosensores.

Dado que el circuito se encuentra separado en dos pisos, se realizó la descripción de cada uno de los pisos por separado en dos sistemas. Cada uno de estos sistemas posee tres bloques, uno con toda la lógica otro para realizar las interconexiones, y un último bloque para los bloques de relleno. La idea de separarlo en tres bloques, radica en que la descripción del circuito en VHDL contenga las estructuras que sirven

para realizar la interconexión entre los dos circuitos y las estructuras de relleno para cumplir con las reglas de diseño. Al sintetizar el circuito hay que tener en cuenta que algunos bloques no están conectados y el software que analiza y sintetiza la descripción puede eliminarlos dado que no cumplen ninguna función lógica.

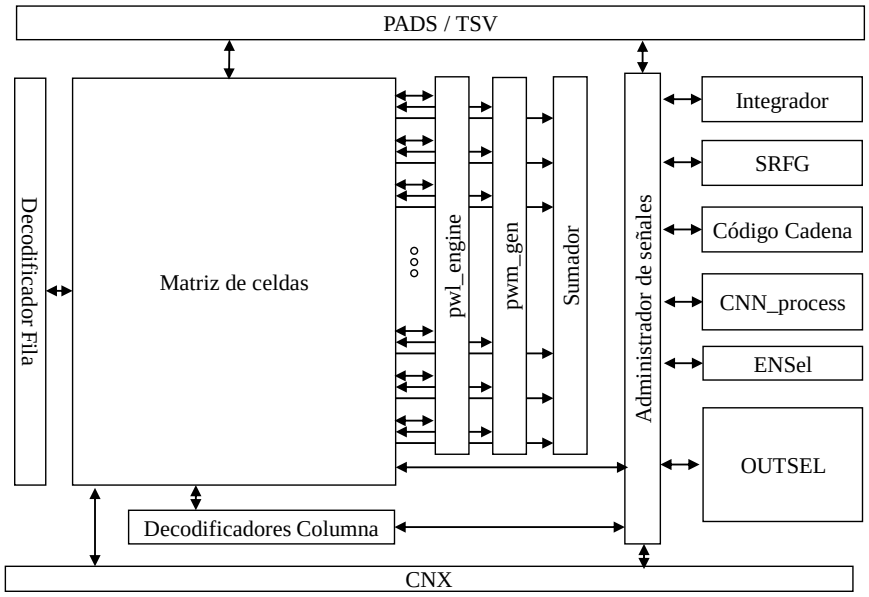


Figura 6.9: Diagrama en bloques de las estructuras colocadas en el piso superior.

La ubicación y distribución de los bloques circuitales se realizó teniendo en cuenta el cumplimiento de la densidad mínima de TSV de polisilicio y de metales, para mantener la integridad estructural de la oblea de silicio. La distribución de los bloques que componen a la celda se realizó de acuerdo a la siguiente estrategia. Se separó la celda en dos grupos de circuitos que tengan aproximadamente el mismo tamaño, y que no tengan muchas conexiones entre sí. Luego el grupo que contenía el fotosensor fue colocado en el piso superior, y el otro en el piso inferior. En la Fig. 6.7 se detalla la estructura de las celdas y la ubicación de sus partes en cada uno de los pisos. La cantidad de líneas de comunicación entre un piso y otro es de 8 señales, siete del contador y una del conversor AD.

Los bloques extras que posee el circuito también se separaron en dos grupos, los que tienen acceso directo a los registros de las celdas se colocaron en el piso inferior, y los que comparten información a través del bus interno, se colocaron en el piso

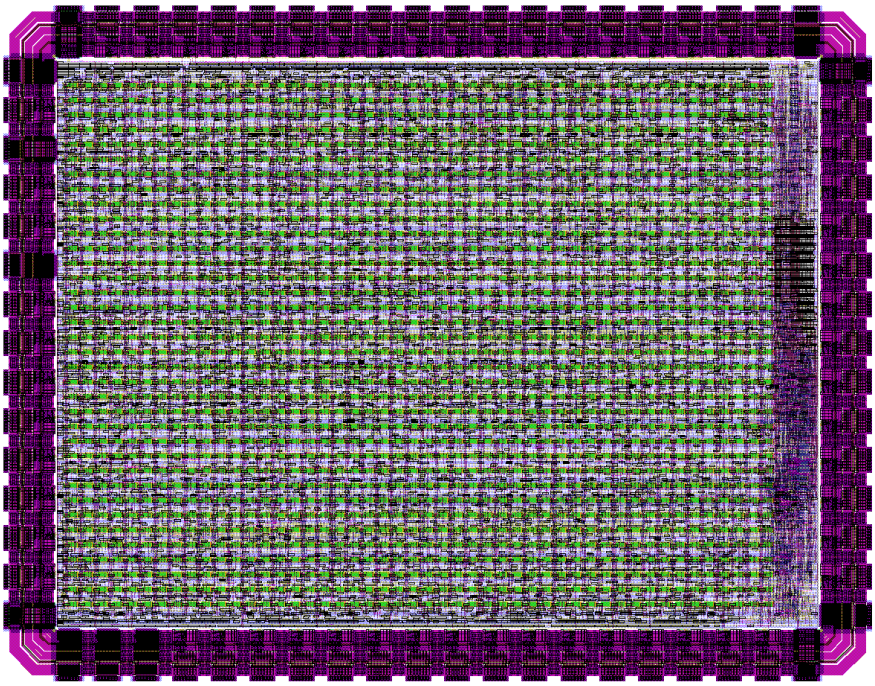


Figura 6.10: Distribución de los bloques y layout del piso superior.

superior.

6.4.1. Piso Superior

En el piso superior se encuentran los PADS, que son las estructuras de conexión con el encapsulado. Este piso recibe todas las señales y la alimentación, muchas de las cuales deben ser enviadas al piso inferior. Este piso se encuentra expuesto y por el cual se hacen las interconexiones con el encapsulado. Por eso en este piso se, los sensores de luz, los pads, las estructuras de protección electroestáticas, el manejador de señales, etc. Todas las estructuras colocadas en el piso superior se muestran en la Fig. 6.9. Una imagen de la distribución de los bloques en el chip y la máscara del piso superior se muestra en la fig 6.10

6.4.2. Piso Inferior

En el piso inferior se colocó el microprocesador, el correlador, el multiplicador, el divisor, y la máquina de procesamiento automático de la CNN. Este piso no se encuentra expuesto a la luz, y debe recibir todas las señales y la alimentación del piso superior. Para pasar alimentación de un piso a otro, se usaron más de 150 conexiones, distribuidas por todo el borde del chip.

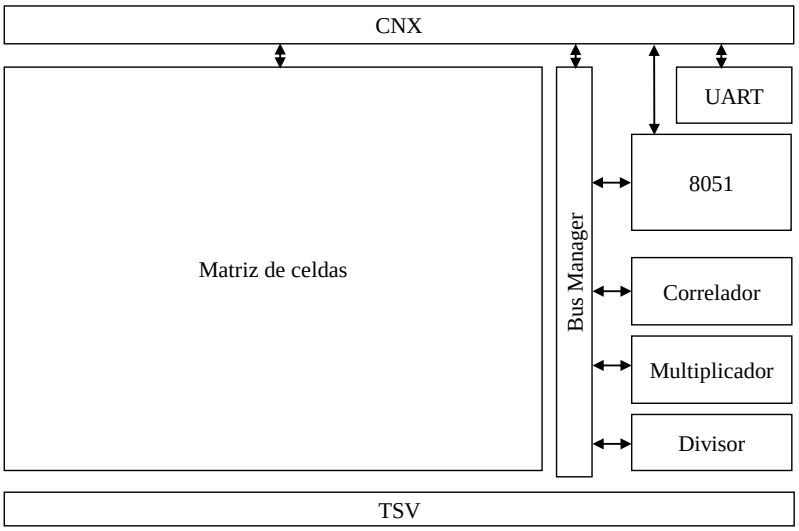


Figura 6.11: Diagrama en bloques de las estructuras colocadas en el piso inferior.

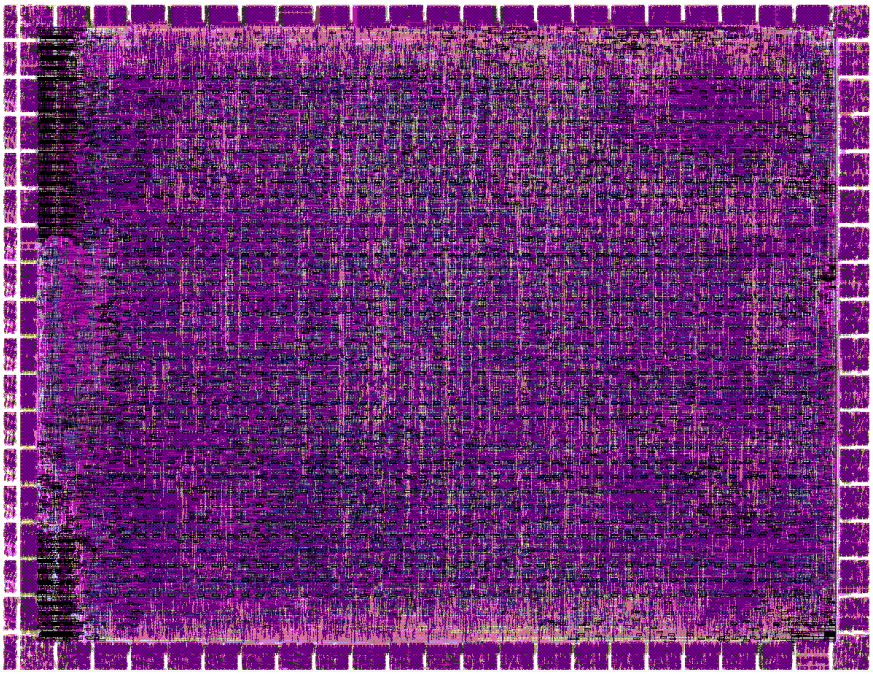


Figura 6.12: Distribución de los bloques y máscara del piso inferior.

Puerto	Función
0	Señales de BUS
1	Dirección de columna
2	Dirección de fila
3	Señales de control
4	Señales de reloj
5	Señales de configuración
6	Señales de Habiliitacion
7	Selección de Bus Interno y Externo

Tabla 6.2: Lista de Puertos y su funcion.

6.5. Funcionamiento

En esta sección se describen las señales que controlan el circuito, se explican sus funciones y se ilustra paso a paso como realizar algunas de las tareas básicas del circuito. También se muestran resultados de simulaciones y testeos realizados previos a la fabricación.

6.5.1. Señales

El circuito completo puede ser manejado por 62 señales, las cuales se enumeran en la Tabla. 6.3. Estas señales generan las habilitaciones y configuraciones necesarias para el control de circuito completo. Estas señales se han organizado en ocho puertos de ocho bits, para un mejor control con microprocesadores de ocho bits. La división de las señales en función de la tarea que cumple se detalla en la Tabla 6.2. También se cuenta con señales de control del microprocesador y señales de direcciones y datos que se usan para ejecutar programas desde una memoria externa. Debido a que esta es una gran cantidad de señales para el control externo del circuito se han mutiplexado, las de configuración, habilitación, Selección de Fila y Columna.

Todos los bloques internos del circuito se conectan a un Bus Interno, al cual todas las estructuras tienen acceso de lectura y escritura. El bloque que toma el

FGBDData	Valor de entrada de funcion de memoria y datos
COL Externo	Valor de columna ingresado externamente
ROW Externo	Valor de fila ingresado extrernamente
InitProc	Realizar Procesamiento de imagen
InitCnt	Todas las celdas cuentan
InitPwm	Todas las celdas realizan el PWM
InitChain	Ejecución del código cadena
Load1	Habilitación carga dato 1
Load2	Habilitación carga dato 2
Cbusb	Selección de Bus o Contador para la entrada a la celda
LX	Carga valor en registro X
LU	Carga valor en registro U
LP	Almacena el valor del PWM en los registros Fpwm y Gpwm
CCLK	Reloj del contador
CRST	Señal de reinicio del contador
RST	Señal de reinicio
BordeU	Valor de los bordes correspondientes al registro U
BordeX	Valor de los bordes superiores e izquierdos del registro X
BordeXB	Valor de los bordes inferiores
BordeXR	Valor de los bordes derechos
Mask	Selecciona si la vecindad para la funcion X proviene de los vecinos o del registro X
SelVecF	Selección de dirección de vecindad
FogIn	Selección de operación entre funciones
EnSel	Habilitación de los distintos bloques de las celdas
RstPxl	Inicializar Pixel
RstSample	Descargar muestreo de pixel
Sample	Muestrear pixel
SelBusOut	Seleccionar la señal del Bus de salida
SelBusIn	Seleccionar la señal del Bus interno

Tabla 6.3: Lista de señales con las que se controla el chip.

SelBusIn	BUSint
0	RegF
1	RegG
2	CountH
3	CountL
4	MultH
5	MultL
6	DivH
7	DivL
8	CorH
9	CorL
10	LongChain
11	Sum
12	FromM
13	ToM
14	SerialPort
15	BUSExt

Tabla 6.4: Selección de señales del bus interno.

control del bus (escribe) se selecciona a través de la señal $SelBusIn$. La tabla 6.4 indica que señal se encuentra en el bus interno en función de la señal $SelBusIn$.

El circuito cuenta con un solo reloj y cada uno de los bloques posee su propia habilitación. El bloque EnSel se encarga de generar las habilitaciones correspondientes para habilitar cada uno de los bloques. La selección de cada uno de los bloques se realiza a través de la señal $HabEn$, la Tabla 6.5 muestra la habilitación de los bloques en función de la señal de entrada $HabEn$.

Algunos de los bloques necesitan habilitaciones internas para seleccionar partes

Habilitaciones	
HabEn	Bloque habilitado
0	Ninguno
1	PWMing
2	Adding
3	Chaining
4	Processing
5	Reading
6	EnDIV
7	EnMUL
8	EnCORR
9	LoadBRG
A	LoadFG
B	HabAnalog
C	HabCmp

Tabla 6.5: Habilitación de los diferentes bloques del chip.

BLOQUE	LOAD1	LOAD2
FG	Carga de Valor F	carga de Valor G
DIV	Divisor	Dividendo
MULT	Multiplicador A	Multiplicador B
CORR	Carga registro V	Carga Registro T

Tabla 6.6: Habilitaciones internas de los bloques.

de ellos. El multiplicador debe cargar los dos valores a multiplicar, el divisor debe cargar el divisor y el dividendo, el correlador debe cargar sus dos valores en los registros V y en los registros T. La Tabla 6.6 muestra los valores de las señales Load1 y Load2 que habilitan las diferentes partes de cada uno de los bloques.

El circuito posee un solo bus de salida con el que se pueden seleccionar distintos nodos internos del circuito. La selección del nodo a leer se realiza por medio de la entrada *SelBusOut*. Todas las posibles salidas se muestran en la tabla 6.7.

El microprocesador posee ocho registros internos (de la dirección 00h a la 08h) los cuales estan mapeados a los ocho puertos que controlan el circuito. Al seleccionar el micro como controlador del circuito, el administrador de señales conecta los ocho puertos de entrada del circuito a la memoria del micro, tal como lo muestr ala Tabla 6.2.

6.5.2. Tareas Paso a Paso y Simulaciones

En esta sección se describe paso a paso algunas de las tareas que ejecuta el imager. mostrando las simulaciones realizadas con el ModelSim.

SelBusOut	BusOut
1	BUSINT
2	ROW
3	HPWM
4	HCNT
5	BusData
6	Enabling
7	Proc
8	CLKs
9	ChainDir
A	Chain
B	Pi3
C	Pi5
D	Counter0,0
E	Fog
F	F(7..0)
0	MICRO3

Tabla 6.7: Selección de señales del bus de salida.

Procedimiento de Lectura y escritura de una imagen o de un valor en un registro

Para almacenar una imagen en el arreglo, se debe direccionar una por una las celdas, y con el dato a guardar en el bus de entrada, aplicar una señal de CLK habilitando el registro correspondiente. Para su lectura, debe ser direccionado, y la salida muestra el valor del registro. La Fig. 6.13 muestra la secuencia de señales para lectura y escritura.

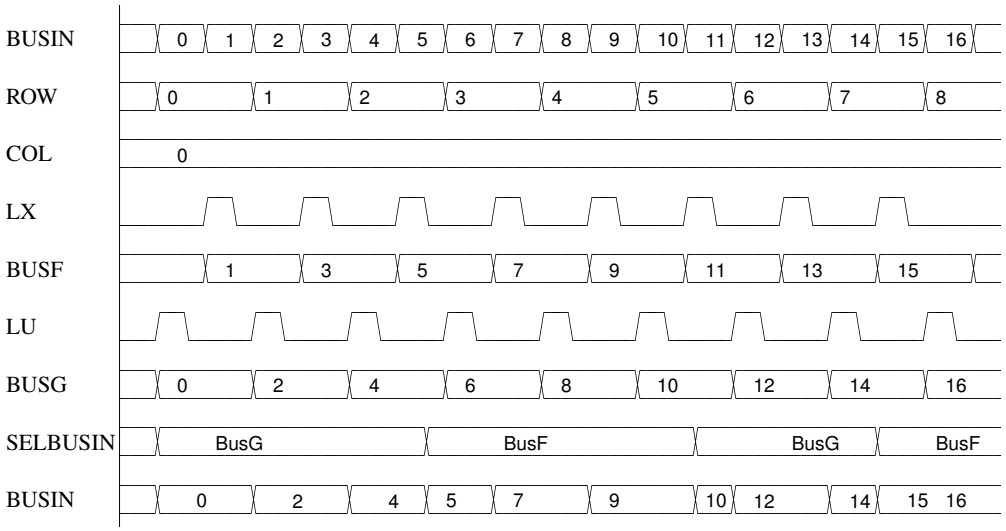


Figura 6.13: Secuencia de lectura y escritura de los registros.

Procedimiento de Procesamiento de imagen

Para realizar el procedimiento de cálculo, como fue explicado previamente, se necesitan 2 pasos, primero evaluar el valor de la celda con la *Rampadeprograma*, obtener el valor de la función e integrarlo. Dado que el generador pwm y el acceso a la memoria esta compartido por las celdas, estos dos pasos no se pueden hacer en la misma celda simultáneamente, porque los valores de todos los vecinos de la celda deben estar actualizados para obtener la dirección del vértice. Es por eso, que hay 2 habilitaciones, una para acceder al comparador, y otra para acceder a la memoria. Una vez que todos los vecinos de la celda fueron actualizados, se accede al

multiplexor que lee la memoria, y se integra el valor del vértice. La Fig. 6.14 muestra un diagrama de cómo es la secuencia.

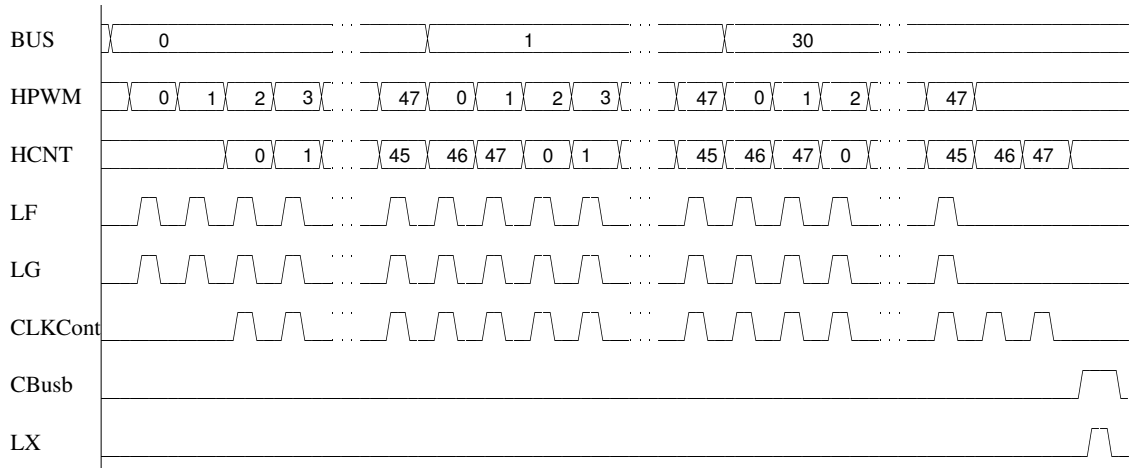


Figura 6.14: Secuencia de señales para realizar el procesamiento S-CNN

Para realizar el procesamiento de la imagen, como primer paso se debe cargar la función que se desea aplicar (tanto F como G). Luego se selecciona el tipo de vecindad a la cual se le va a aplicar la función. Luego se aplica la *Rampadeprograma* (cada paso puede constar de uno o de dos ciclos de reloj, teniendo en cuenta si se va a utilizar una función o las dos). Como último paso se debe cargar el valor del contador en un registro. A continuación se explica con más detalle los pasos para realizar estas tareas.

1- Carga de la Función:

Como primer paso se deben cargar las funciones F y G en el registro serie. Una vez cargados los valores se almacenan en los latches, para que los buffers transmitan la información a las 1536 celdas.

2- Selección de Vecindad y Operación entre F y G:

Con las señales $SelVecF$ y $SelVecG$ se selecciona la vecindad que van a tener los registros F y G. Con la señal FoG (es una señal de 2 bits) se selecciona que tipo de fusión se aplicara a la

3- Rampa de programa:

Para cada paso de la *Rampadeprograma* hay que calcular el valor de la señal

PWM de cada celda, y una vez que se tiene el valor de todos los vecinos, se obtiene el valor de la función. No se puede realizar la comparación y leer la función en la misma celda, dado que todos los vecinos tienen que tener actualizado su señal pwm. La obtención de la función se hace una vez que todos los vecinos de las celdas hayan realizado la comparación, en la forma de procesar de este circuito, la señal HPWM debe ir una columna más adelante que HCNT, tal como lo muestra la Fig. 6.14.

4- Almacenamiento del nuevo estado en un registro:

En este paso el valor del contador se almacena en uno de los dos registros de la celda. Con la señal $CBusb$ se selecciona que al contador como entrada de dato de todos los registros y luego con un pulso de reloj en el registro destino se almacena el valor, tal como lo muestra la Fig. 6.14, donde el valor del contador se almacena en el registro X.

La distribución de la memoria, y la precisión de los registros hacen que el circuito procese como máximo en $126 * 2 * 48 + 2 = 12098$ donde 126 son los ciclos de la *Rampadeprograma* para una imagen de 7 bits, 2 ciclos de reloj para aplicar la función F y la función G, 48 ciclos para realizar la generación de la señal PWN, y dos más para realizar la integración de los valores de la función. Como minimo, son necesarios $1 * 48 + 2 = 50$ ciclos de reloj, para una imagen blanco y negro a la cual se le aplica una sola funcion(F o G).

Uso del sumador y del integrador

Al seleccionar una celda para ser leída, todas las celdas de la columna, escriben el valor del registro en el bus de su fila correspondiente, y el sumador asincrónico, en su salida, muestra la suma de todos los valores.

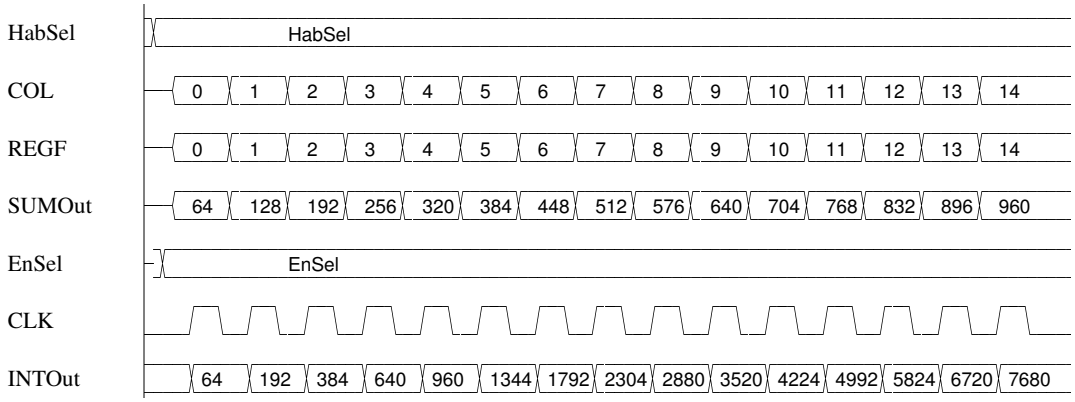


Figura 6.15: Simulación de uso del integrador

Uso del multiplicador y del divisor

Para usar tanto el divisor como el multiplicador se necesitan cargar los dos datos de 8 bits, de sus entradas, multiplicando y multiplicador, dividendo y divisor. Esto se hace a través de la señal *loadA* y *LoadB*. La Fig. 6.16 muestra la secuencia de pasos para la utilización del multiplicador.

Uso del Microprocesador

Para usar el microprocesador es necesario que la entrada microExtb este en el estado lógico alto. La memoria de programa en este caso no se integró, lo que significa que debe estar conectado al chip por medio de los pines definidos para ese fin.

Las Fig 6.17 y 6.18 muestran pantallas de las simulaciones realizadas con el software ModelSim.

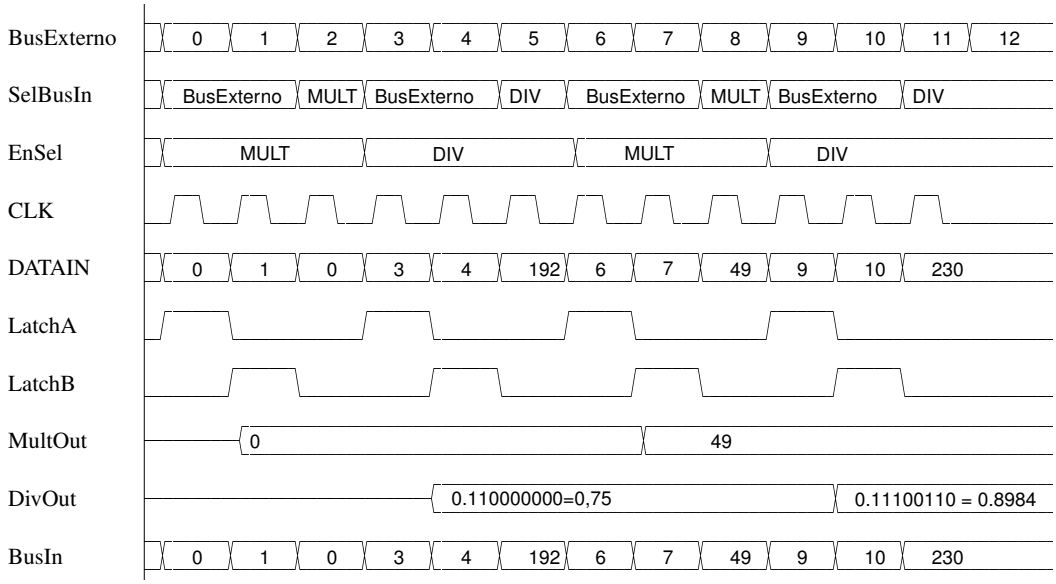


Figura 6.16: Simulación de uso de multiplicador y divisor.

6.6. Conclusion

En este capítulo se ha descripto la realización de un circuito integrado para la implementación de un Imager S-CNN, en un proceso 3d. Este proceso cuenta con 2 obleas interconectadas, con un tipo de conexión flip chip. La arquitectura de las celdas fue seleccionada para minimizar la complejidad del arreglo, separando las estructuras en cada uno de los "Tiers"de manera de minimizar la cantidad de conexiones, dado que se utilizaría un software de conexionado automático. Se incorporaron varias estructuras extras para aumentar la capacidad de procesamiento y junto con un microprocesador con 2 sistemas de comunicación con el exterior, de manera de poder interconectar varios circuitos integrados trabajando en paralelo. Si bien el integrado se encuentra todavía en fabricación, las simulaciones en VHDL, y los testeos de las estructuras en la FPGA muestran un funcionamiento correcto de la estructura. La tabla 6.8 muestra un resumen de las características del Imager.

Imager	
Tecnología	CMOS $130nm$ 5 metales 1 Poly
Arreglo	48×32
Tamaño del integrado	$2mm \times 2,5mm$
Imágenes	Escala de Grises
Precisión	$\sim$ 7 bits
Memorias de imagen	2
Vecindad	de 4 (Seleccionable entre + y x)
Tamaño celda	$25\mu \times 25\mu m$
Fill Factor	25 %
Estructuras Extras	Sumador de valores de Columna, Integrador, Multiplicador, Divisor, Correlador, Generador de código cadena, Microprocesador 8051
Velocidad de trabajo	100Mhz
Mínima cantidad de clocks para procesar	50
Máxima cantidad de clocks para procesar	12098

Tabla 6.8: Características de la cámara inteligente

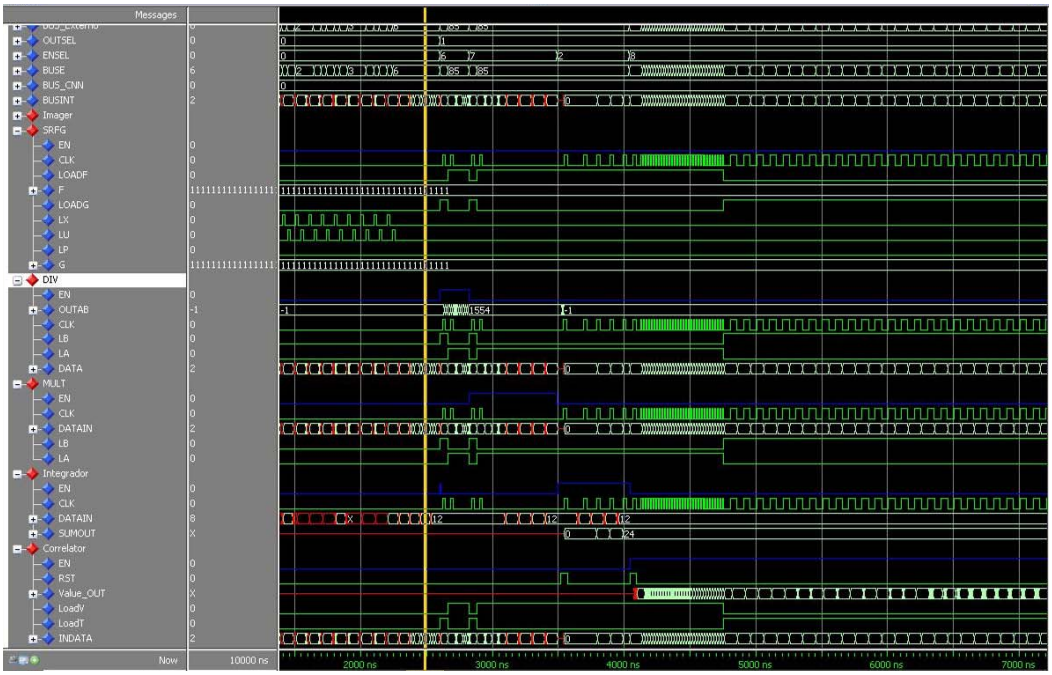


Figura 6.17: Pantalla del ModelSim mostrando resultados de la simulación.

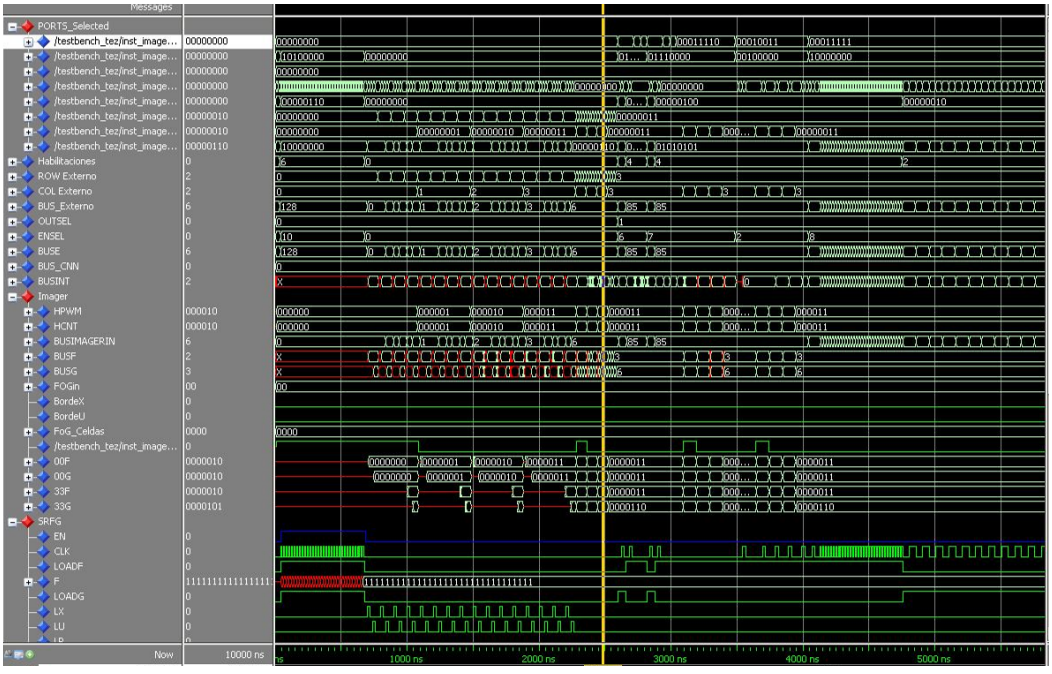


Figura 6.18: Pantalla del ModelSim mostrando resultados de la simulación.

Capítulo 7

Sistemas de Verificación de Imagers

7.1. Introducción

La comprobación del funcionamiento de los circuitos integrados y la medición de variables como consumo y velocidad de trabajo son un paso fundamental para asegurar que un circuito funciona correctamente y que cumple con las especificaciones de diseño. Las señales relevantes, las estructuras de testeo y el procedimiento de medición deben planificarse en la etapa de diseño del circuito (DFT), para asegurar la observabilidad de señales internas que garanticen el funcionamiento previsto. En un circuito tipo CNN, la complejidad es grande dado que se trata de un sistema con muchas celdas, cada una con múltiples estados internos y entradas de vecinos. Un sistema de medición debe ser confiable, robusto y tener la precisión necesaria para realizar las mediciones. Para esto se debe contar con instrumental adecuado que pueda generar todas las señales necesarias con la precisión adecuada, y que además pueda captar las imágenes para almacenarlas/mostrarlas para luego realizar la verificación del funcionamiento.

En este capítulo se describen los sistemas de verificación que se crearon para realizar las mediciones a los circuitos integrados fabricados, y también se muestran resultados de mediciones realizadas. El capítulo está organizado como se describe a continuación. Se comienza con una introducción de los parámetros más importantes de los imagers, y las variables que se deben medir o calcular. En la Sección 7.2 se describen los sistemas de medición que se crearon para generar y medir las señales

para comprobar el funcionamiento de los circuitos. En esa sección se describen dos sistemas de verificación, el primero es un sistema genérico para testeos de circuitos integrados de señal mixta y el segundo es un sistema de verificación basado en una FPGA de la familia Xilinx®. En la Sección 7.3 se describen las mediciones realizadas a los imagers fabricados entre los cuales se encuentra el S-CNN introducido en el Cap. 2, el circuito fabricado con una tecnología 3D del MIT Lincoln Lab, y dos circuitos de prueba con celdas aisladas fabricadas en una tecnología de $0,35\mu m$ (una diseñada con lógica de pre-carga y otra con lógica estática). Por último se muestran algunos datos del chip fabricado cuyo diseño y arquitectura se explican en el Cap. 5.

7.1.1. Verificación de circuitos

Para realizar la verificación de los circuitos integrados es fundamental contar con un sistema para generar y captar señales eléctricas. También es importante la forma en la cual la información obtenida es mostrada a la persona que realiza la verificación.

Antes de comenzar a realizar el testeo, se debe tener completo conocimiento del funcionamiento del circuito. Se debe conocer cuál es la tarea de cada bloque, qué función cumple cada una de las entradas, cuáles son las señales que se generan dentro y así diseñar un sistema que genere las señales adecuadas para una correcta verificación. Una vez que se ha realizado el sistema de verificación, se debe comprobar su correcto funcionamiento antes de conectar el chip a medir. Para esto se debe verificar el envío y la recepción de todas las señales, en cada uno de los pines y verificar la comunicación con la computadora. Dado que en un imager, el circuito debe estar expuesto a la luz, se debe disponer de un encapsulado de cavidad abierta. Este tipo de encapsulados dejan también expuestas las líneas metálicas que vinculan los pads del circuito integrado y los pads del encapsulado. Esto requiere un manejo cuidadoso al manipular el chip y/o la placa. Para proteger el conexionado del encapsulado se debe crear una cubierta protectora. Luego de colocar el chip a testear, y antes de comenzar con el testeo, se debe asegurar que todos los pines del chip están conectados correctamente a los puertos de la placa correspondientes, y que se encuentren eléctricamente vinculados (que no haya ningún corto o circuito abierto) de manera

de estar seguro que las señales que se generan llegan correctamente al circuito.

Una vez que se tiene la certeza que las señales generadas llegan al chip, que las señales leídas son las que enviadas por el chip, se puede comenzar con las mediciones de funcionalidad y eficiencia. La funcionalidad debe abarcar tanto las partes digitales como las analógicas. Para comprobar el correcto funcionamiento de las partes digitales, se debe verificar el funcionamiento de los registros de las celdas, los comparadores digitales, la memoria de programa, y las estructuras extras. Se debe probar el funcionamiento del arreglo CNN procesando imágenes con diferentes funciones y operaciones entre funciones. Para comprobar el correcto funcionamiento de la parte analógica se debe verificar el funcionamiento del fotosensor, la relación entre la corriente inversa del fotodiodo y la intensidad de luz, y el comparador utilizado para realizar la conversión AD.

Con respecto a las variables de eficiencia se mide el consumo de potencia del chip y de los pads, el consumo de potencia estático y dinámico a diversas frecuencias y tensiones, para poder calcular el consumo por celda, por instrucción y por procesamiento de una imagen completa. También se debe verificar en que rangos de frecuencia / voltaje el circuito funciona correctamente.

Mediciones a realizar con un circuito integrado:

- *Consumo:* Consumo del chip total, consumo de cada celda, consumo por operación, consumo en función de la velocidad, consumo en función de la tensión, consumo en función de la tarea que se esté realizando, etc.

- *Velocidad de trabajo:* Máxima velocidad de funcionamiento, velocidad de captura de imágenes, zona de funcionamiento en función de la tensión y la frecuencia (Grafico shmoo).

- *Funcionamiento de los registros:* Verificar el funcionamiento de cada uno de los registros de todas las celdas y de la memoria de programa.

- *Captura de Imágenes:* Obtener el patrón de ruido fijo, medir el tiempo de integración en función de la luz, cantidad de imágenes por segundo que se pueden obtener, etc.

- *Procesamiento de imágenes:* Realizar el procesamiento de las imágenes con diferentes funciones, y verificar que la salida sea correcta.

7.2. Sistemas de Verificación

A los fines de llevar a cabo la verificación funcional y la medición de variables de eficiencia de los chips fabricados se han realizado dos sistemas de verificación. Cada uno de estos sistemas posee un área para conectar el dispositivo a verificar, una conexión a la computadora para recibir y enviar información, rutinas de software que se ejecutan en el mismo sistema y una interfaz gráfica en la computadora que permite enviar, recibir y visualizar señales de una manera simple y automatizada. El primer sistema mostrado aqui, es un sistema genérico para la verificación de circuitos integrados de señal mixta, digital y analógico, el cual consta de una placa con capacidad de manejo de 32 señales digitales configurables como entrada, salida o bidireccionales y 10 puertos analógicos de entrada y 10 de salida. El segundo sistema es un ambiente de verificación para ser utilizado con placas de desarrollo que contengan una FPGA de la familia Xilinx®. Al trabajar con lógica programable, el circuito solo envía y recibe señales analógicas, lo que deja al usuario la necesidad de colocar un conversor A/D o D/A para el manejo de señales analógicas. Este circuito tiene la posibilidad de manejar hasta 256 puertos de ocho bits (2048 señales). La cantidad de señales dependerá de la placa de desarrollo con la que se cuente para conectar el circuito a testear.

7.2.1. Sistema Genérico

Hardware

Este sistema fue diseñado como una plataforma flexible que permitiese el testeo de distintos tipos de circuitos. El sistema se basa en un microprocesador de la familia PIC 18LF6680 de Microchip, dos CPLD XC2C64 (también se pueden colocar los XC2C32A) de Xilinx, un conversor analógico digital LTC1660 y un convertidor de niveles de señal MAX233 para la comunicación serie. La placa cuenta con dos puertos bidireccionales de 16 bits, ocho puertos analógicos de entrada, ocho puertos

analógicos de salida y un puerto de comunicaciones RS232. La Fig. 7.1 muestra un diagrama en bloques del sistema de medición.

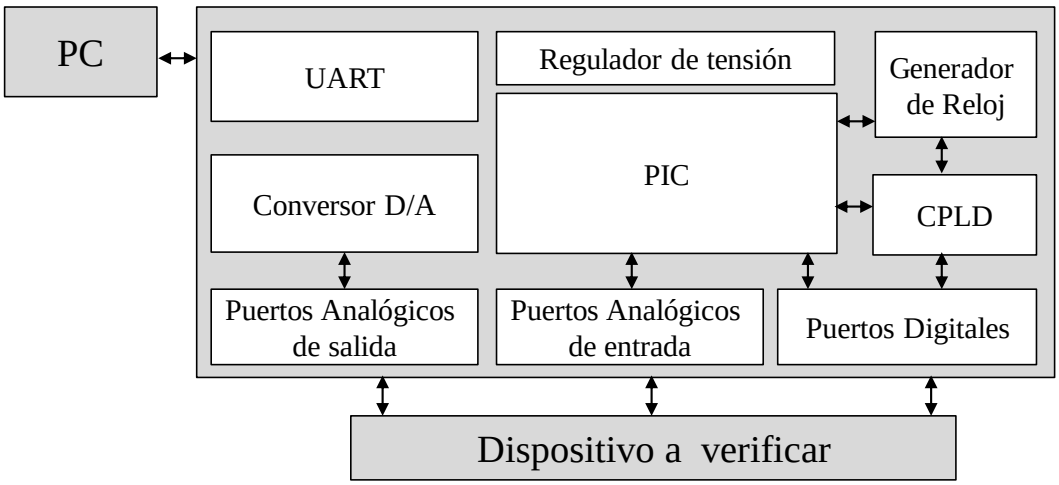


Figura 7.1: Diagrama en bloques de la placa de verificación genérica

Una de las características a tener en cuenta al diseñar sistemas de testeo para circuitos integrados es la variación de tensiones de alimentación, lo cual depende fundamentalmente del proceso usado. Esta variación puede ir de 5V para chips de tecnologías de 0.5um a 2um, hasta 1V para tecnologías de 65nm. Es importante prever el sistema para que pueda acomodar un rango útil de tensiones de alimentación. En este caso, se dispusieron dos CPLD como lógica intermedia entre el microprocesador de control y el chip, tanto para acomodar los niveles de tensión, como para agregar más lógica al sistema. Los niveles lógicos de tensión del sistema placa varían desde 1.2V hasta 3.5V (estos son los niveles admitidos por el CPLD). La señal de reloj llega hasta el micro y a través de una llave se puede hacer llegar a los CPLDs. El microprocesador puede correr hasta 40Mhz, enviar y recibir señales a una frecuencia de 10Mhz. Los CPLDs pueden funcionar a una velocidad máxima de 300Mhz. La Fig. 7.2 muestra una fotografía del sistema completo donde se puede observar un área de prototipado para realizar conexiones cableadas (wire wrap) con el chip a testear..

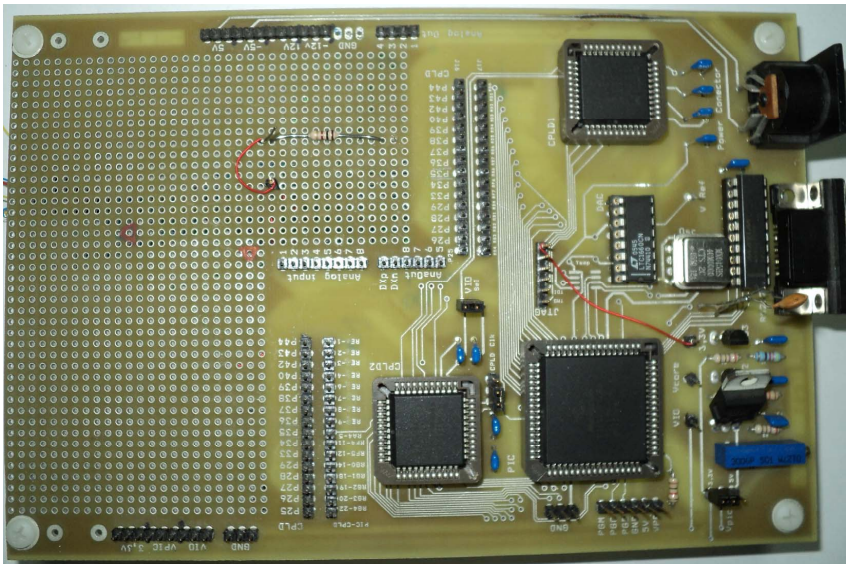


Figura 7.2: Fotografía de la placa de verificación genérica

Software

Software en la placa:

El microprocesador posee funciones básicas que ejecuta al recibir comandos desde la computadora. Este micro se encuentra conectado a los dos CPLDs y a los conversores D/A, y se encarga de generar todas las señales para controlar estos dos circuitos, mediante cuatro comandos básicos: leer puerto digital, escribir en puerto digital, leer puerto analógico, escribir en puerto analógico. También es posible agregar funciones específicas para realizar rutinas especiales. Además de los comandos básicos se han agregado otros como una señal con temporizado, la generación automática de rampas y trenes de pulsos. La programación se puede modificar a través de del programador de Microchip, y se pueden realizar rutinas nuevas en assembler. Dentro de las posibilidades que brinda el sistema, se puede describir el comportamiento de los CPLDs, los cuales normalmente actúan como buffer de salida y entrada, utilizándose para acomodar los niveles de tensión, pero también se pueden utilizar para agregar más lógica, modificar el ruteo de las señales, almacenar datos o crear máquinas de estados. Dado que el micro solo puede operar señales a 10Mhz, se puede usar el CPLD para generar señales a mayor frecuencia.

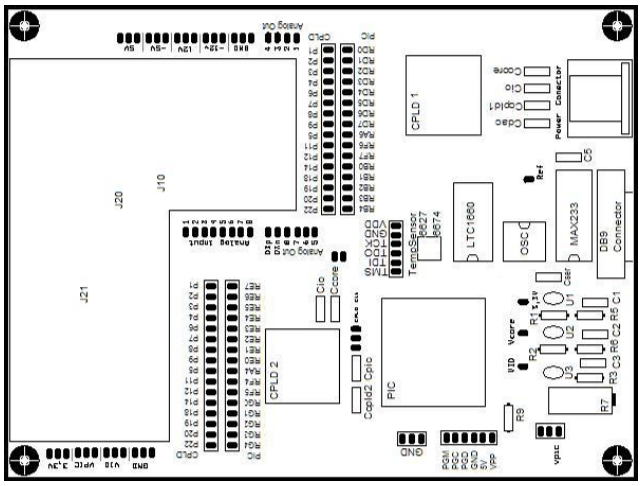


Figura 7.3: Distribución de componentes en la placa de verificación genérica.

Software en la PC:

La computadora posee rutinas en MatLab® para enviar comandos al circuito. Entre ellas se encuentra, leer puerto digital, escribir en puerto digital, leer puerto analógico, escribir en puerto analógico. Si se agregan más comandos al chip, solo se deben crear más rutinas para enviar el comando a la placa. La generación de patrones de señales se realiza programando los vectores en MatLab y luego se envían los comandos respectivos con los valores de los vectores a escribir. Cada comando enviado actualiza la salida de la placa correspondiente, así que la velocidad de la conexión del puerto serie define la velocidad con la que las señales se actualizan. Si se conoce el patrón de señales a generar se puede programar directamente el controlador para que las genere a una velocidad de 10Mhz.

7.2.2. Sistema de verificación basados en FPGA

Hardware

Este segundo sistema de prueba se diseñó para generar y muestrear mayor cantidad de señales o señales de mayor longitud de palabra, dado que el sistema de verificación genérico solo tiene dos puertos de 16 bits. El diseño de esta estructura se muestra en la imagen 7.4. En este diseño, la FPGA se encarga de generar todas las

señales a través de comandos enviados por la computadora. El sistema de medición utiliza el microcontrolador Picoblaze® de Xilinx -razón por la cual solo puede ser utilizado en las FPGAs de esta marca. El controlador consta de un procesador que posee 256 puertos de entrada y salida y un conjunto de más de 20 instrucciones. El micro solo direcciona los datos y envía una señal de lectura o escritura, externamente se ha creado un multiplexor para la lectura y un arreglo de registros para los puertos. El sistema actual consta de ocho (direccionados entre el 240 y el 247) puertos de escritura y seis (direccionados entre el 240 y el 245) de lectura de ocho bits cada uno. Junto con el controlador se instancio una UART para comunicar el sistema con la computadora y enviar y recibir información. Debido a que este circuito está sintetizado en una FPGA, es posible colocar más de un controlador, para trabajar en paralelo, o para que cada uno realice tareas específicas. Gracias a la gran versatilidad y conectividad de las FPGAs, se pueden realizar modificaciones y mejoras de manera muy sencilla y rápida. En el laboratorio se cuenta con dos placas de desarrollo con circuitos de Xilinx. A continuación se describe cada una de ellas.

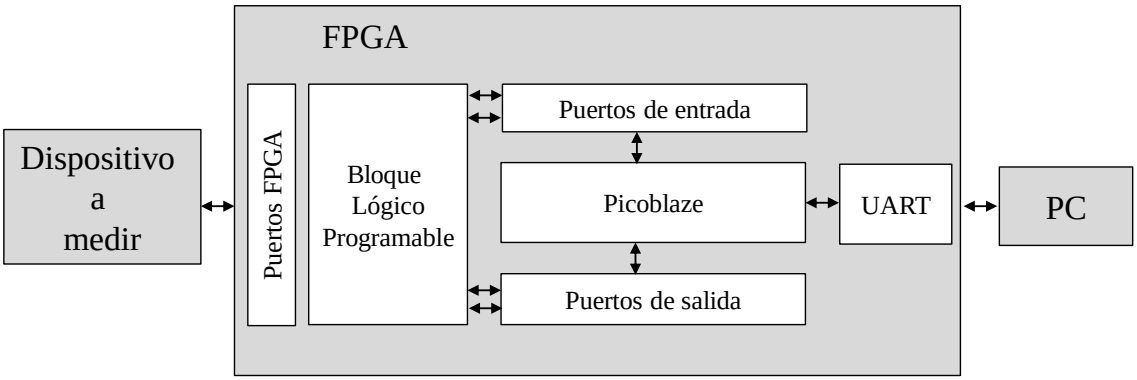


Figura 7.4: Diagrama en bloques de la placa de mediciones.

- Kit Digilent:

  Esta placa posee tres puertos con un conector tipo DIP, de 40 bits cada uno. No en todos los puertos los 40 bits están disponibles porque los pines están compartidos con otras características de la placa, como alimentación, señales de reloj, acceso a la memoria, etc.

En total se pueden manejar unas 60 señales. La comunicación a la computadora se realiza a través de un puerto serie con una velocidad máxima de transferencia de datos de 115200 baudios. La placa tiene un reloj de 50Mhz, y dentro el circuito puede correr a 100Mhz. La velocidad con la que se pueden enviar señales al exterior es de 50Mhz.

- Kit Memec:
  Esta placa posee dos puertos de 80 bits en los cuales se puede colocar un conector. La comunicación a la computadora se realiza a través de un puerto USB con una velocidad máxima de transferencia de datos de 1M baudios. La placa tiene un clock de 75Mhz, y dentro el circuito puede correr a 150Mhz. La velocidad con la que se pueden enviar señales al exterior es de 75Mhz.

Software

Software en la placa.

Estos sistemas corren programas de assembler, los cuales están constantemente leyendo el puerto serie para ver si llega algún comando desde la PC. Una vez que llega algún comando, el micro ejecuta el comando que se ha enviado. Se cuenta con comandos básicos de escribir datos en un puerto, leer datos de un puerto o recibir datos del estado del controlador. También se cuenta con comandos más complejos como correr la rutina completa de procesamiento de una CNN. La tabla 7.1 muestra la lista de comandos que se utilizaron para realizar las mediciones del chip de 90nm, descripto en el Cap. 5 y cuyos resultados se muestran en la Sección 7.3.4 de este capítulo.

Software en la PC.

La computadora posee rutinas en MatLab® para enviar comandos al circuito. Los comandos se envían codificados por letras, tal como lo muestra la tabla 7.1. Para ejecutar el comando deseado, solo se debe enviar por el puerto serie la letra correspondiente y los parámetros necesarios del comando. Se ha creado una interfaz en MatLab para enviar la información necesaria al circuito para que corra las rutinas de testeo. Si se agregan más comandos al chip, solo se debe enviar la letra del

Comandos MATLAB			
Comando	Parametros		Funcion
W	Port	Data	Escribe Data en Puerto
E	Data		Coloca Data en BUS
L	U	array	Lee Registro U
	X	array	Lee Registro X
	W	array	Lee Registro W
	T	array	Lee Registro W
	S	lee	Lee Cuenta
	F	funcion	LeeFg
S	RowCol		Selleciona RowCol
F	32		Carga la Funcion F
G	32		Carga la Funcion G
T	B	U	Latchea Bus en U
		X	Latchea Bus en X
		W	Latchea Bus en W
	C	U	Latchea Contador en U
		X	Latchea Contador en X
		W	Latchea Contador en W
	T		Latchea T
	F		Latchea Fpwm
	G		Latchea Gpwm
A	Puerto	Valor	Convierte Analogico
D	Puerto		Convierte Digital
C			Clock para contar
I	SelVec+FoG		Carga CONF
P	C		Contar Pixels
	P		Obtengo PWL decelda
	A		Analisis
Y			Envia Y

Tabla 7.1: Lista de comandos en la placa de verificaciones para testear el chip de 90nm.

comando correspondiente. Las Figuras 7.14 7.25 y 7.31 muestran varias interfaces creadas para enviar y recibir señales a través de este sistema.

7.3. Verificaciones de chips fabricados

En este capítulo se describen los ensayos y resultados de cada uno de los chips reportados en los Cap. 5 y 2.3. Además se detallan los resultados de celdas unitarias fabricadas en una tecnológica de $0,35\mu m$ y Se detallan los pasos realizados para comprobar su correcto funcionamiento y se muestran algunos resultados experimentales.

7.3.1. Imager en tecnología 3D del MIT

En esta sección se analiza el caso de la arquitectura realizada en la tecnología 3D del MIT Lincoln Lab, que consta de un arreglo de 14×14 celdas con dos registros de ocho bits y conectividad con 9 vecinos, diseñado en el marco de la tesis doctoral [41]. Este circuito no tiene recursos compartidos, pero cuenta con una máquina de estados programable para el manejo del arreglo de celdas. La memoria de programa (de 512bits) se encuentra fuera del arreglo y se transmite a todas las celdas a través de 2 señales, lo cual demanda 256 ciclos de reloj.

Descripción de la verificación

Medición de consumo estático:

Con el circuito conectado en la placa se procedió en primer lugar a medir el consumo de corriente estática del circuito. Con todas las señales en un nivel cero lógico se midió la curva de consumo estático del circuito. Se utilizó una resistencia de 100Ohm para medir la corriente y se barrió la tensión de alimentación de 0V hasta 1.8V

Verificación funcional

- Memoria de la máquina de estados:

  A continuación se comprobó la memoria de la máquina de estados. Esta memoria es un registro serie, de manera que los datos ingresados salen desplazados

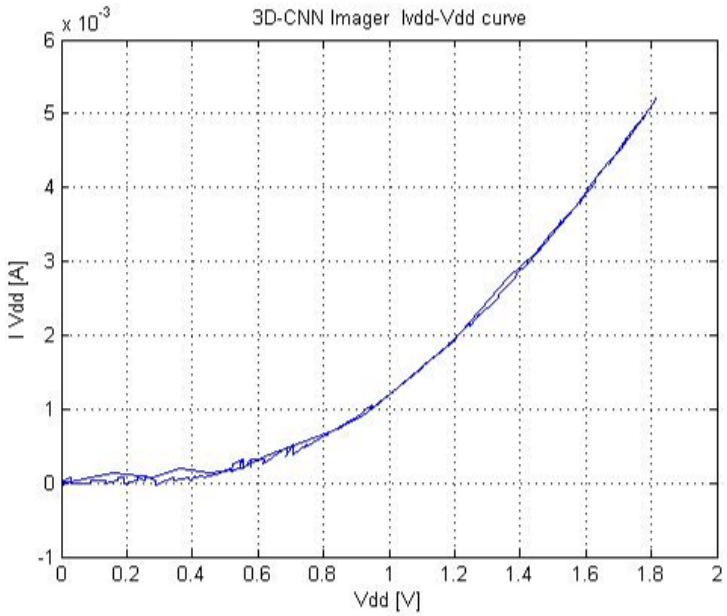


Figura 7.5: Consumo del chip variando la fuente de alimentación.

32 ciclos de reloj. Para la verificación de este circuito, se comprobó la concordancia entre la salida y la entrada. Se utilizaron diferentes vectores de entrada, y la salida fue correcta en todos los casos. La Fig. 7.6 muestra una fotografía de la pantalla del osciloscopio de esta medición.

- Bancos de Memoria:
  Los bancos de memoria fueron verificados de la misma manera, pero el resultado de la comprobación fue incorrecto. Al colocar un cero en el registro serie y activar la señal del reloj, el valor se almacenaba en los primeros 1000 registros, en lugar de hacerlo solo en el primer registro. Este error puede ser producto de que los registros se hagan transparentes transitoriamente o de rebotes en la señal de reloj. En cualquier caso, no se pudo determinar la causa del error mediante mediciones.

- Arreglo de celdas:
  Para verificar el arreglo se almacenó un valor constante en todas las celdas, y se leyó todo el arreglo (manteniendo el valor del bus de entrada en el mismo valor).

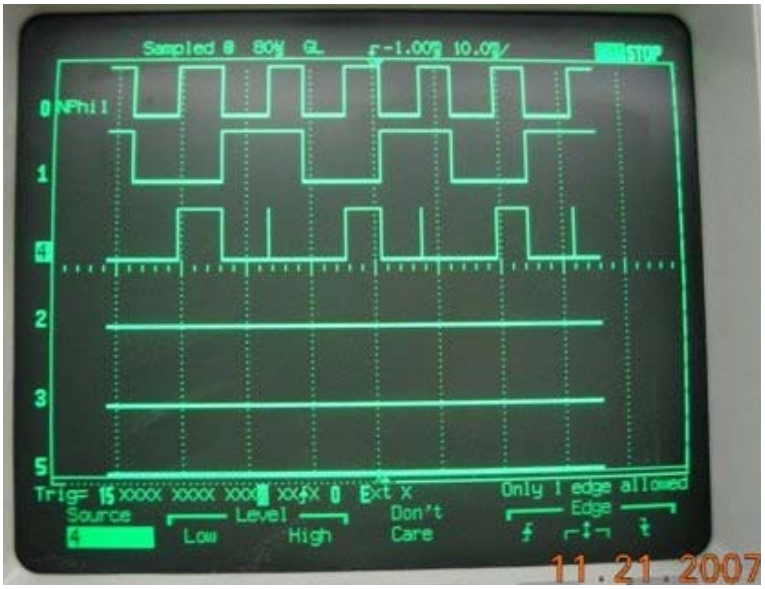


Figura 7.6: Imagen del osciloscopio; la señal superior corresponde al reloj y la inferior a la salida.

En este caso la lectura de todo el arreglo fue satisfactoria. A continuación se almacenó en cada registro el valor de la columna correspondiente ($C_{ij} = j$) y se realizó la lectura de todos los registros. Los resultados, que se muestran en la Fig. 7.7, no fueron satisfactorios.

Al realizar la misma tarea, pero almacenando en las celdas el valor correspondiente a la fila ($C_{ij} = j$) los resultados fueron no satisfactorios. Las mediciones del circuito realizadas con el osciloscopio se muestran en la Fig. 7.8 y el resultado de esta medición en la Fig. 7.9.

Dado que la lectura del arreglo se realiza fijando una columna y variando la fila, puede haber una correlación entre la dirección y el valor almacenado o leído. La Fig 7.9 muestra que los datos concuerdan con la dirección de la columna en vez de concordar con la dirección de la fila. Para verificar la correlación entre la dirección de la celda o el valor del bus al leer la información se realizó un procedimiento de medición que consta de seis pasos, los cuales se explican a continuación:

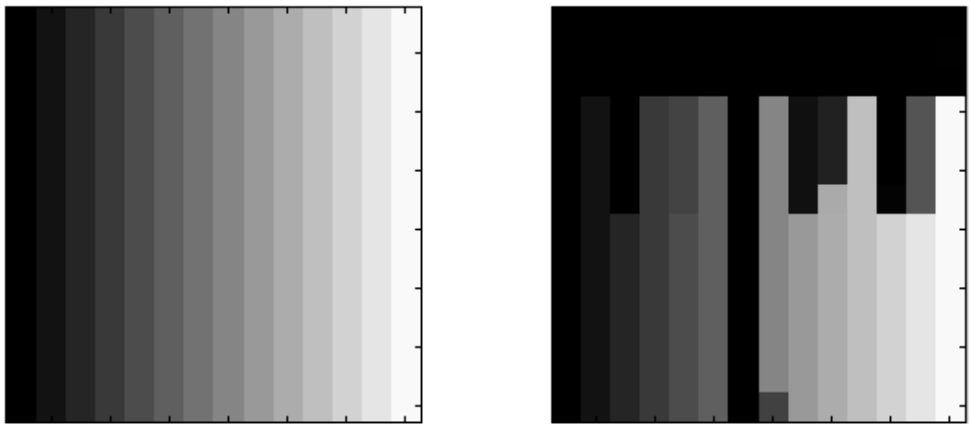


Figura 7.7: Resultado de la escritura y lectura de los registros: (a) Imagen almacenada; (b) Imagen leída.

a) Se coloca en el bus el valor del dato correspondiente D_i (el cual varía entre 0 y 255), se selecciona un píxel y se lee su registro (A)

b) Al terminar de leer la fila, se coloca el bus en cero, y se repite la lectura de todas las celdas de la fila. Esta tarea se repite hasta leer todo el arreglo con un valor de bus D_i y 0. En este caso, las únicas señales que están variando son las habilitaciones de las filas y columnas y el bus de entrada. No se habilitan los registros para ser escritos, de manera que en ambos casos deberían tener el mismo valor.

c) Se coloca el valor D_i en el bus, se selecciona un pixel, se envía la señal de habilitación para la escritura y se lee el valor del registro, manteniendo el valor D_i en el bus. (C)

d) Al terminar de realizar esta tarea con toda la fila, se lee nuevamente toda la fila con el valor D_i en el bus de entrada, (D) (sin enviar la señal de habilitación para escritura).

e) Se leen todos los registros de la fila, con el bus en valor 0 (sin enviar la señal de habilitación para escritura). (E)

f) Al terminar con todo el arreglo, se coloca el valor D_i en el bus, y se lee nuevamente todo el arreglo(F). Para este procedimiento se fija un valor

Figura 7.8: Imagen del osciloscopio: de arriba hacia abajo, dato de salida, señal de habilitación de escritura y bit menos significativo del selector de fila.

de columna, y se barre la fila por completo.

Los resultados de esta medición con un valor de $D_i = 255$ se muestran en la figura 7.10. Allí se puede ver que el circuito almacena valores en las celdas en momentos en los que no debe.

Se verificó la ausencia de ruido en las señales, sobre todo en la señal de habilitación de escritura, dado que los valores en los registros experimentan cambios sin haber sido habilitados. También se verificó la ausencia de movimiento relativo de la tensión de referencia ("Ground bouncing"). De la discusión con otros grupos que también participaron en la corrida experimental del proceso, se observó la presencia de errores similares, que fueron finalmente atribuidos a la tecnología. Debido a esto, se pautó otra corrida con la fábrica.

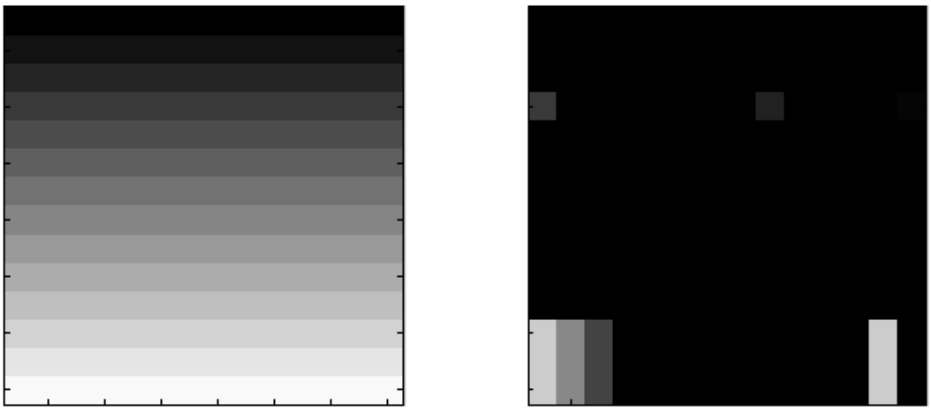


Figura 7.9: Resultado de la escritura y lectura de los registros: (a) Imagen almacenada; (b) Imagen leída.

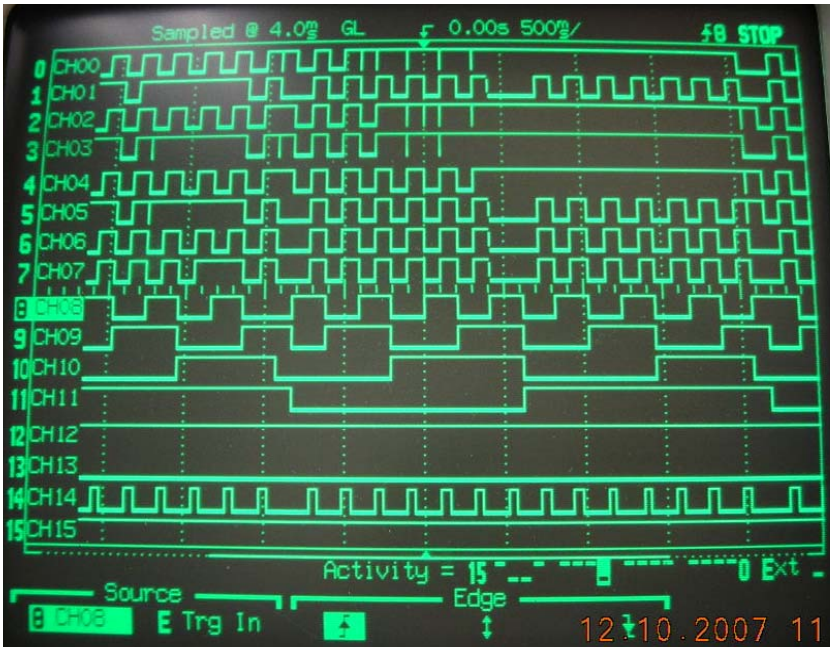


Figura 7.10: Imagen del osciloscopio: Datos de salida (CH0 a CH7), direcciones de filas (CH8 a C11) y señal de habilitación de escritura (CH14).

7.3.2. Imager en tecnología AMI 0,5

Esta sección describe las mediciones realizadas y los resultados obtenidos de la verificación del circuito integrado S-CNN descripto en el Cap. 5 y en los trabajos [14]. Este chip posee dos versiones. Las mediciones de la primera versión se realizaron con una FPGA Spartan3 y las mediciones de la segunda versión se realizaron con la placa de desarrollo genérica. La primera versión del chip fabricado tenía un error, por lo cual las mediciones no se pudieron realizar, para la segunda versión el error fue arreglado las mediciones realizadas y los resultados publicados en [15].

La Fig. 7.11 muestra un diagrama esquemático de la celda. Cada una de las señales parciales se conectan a un flip flop para almacenar el valor.

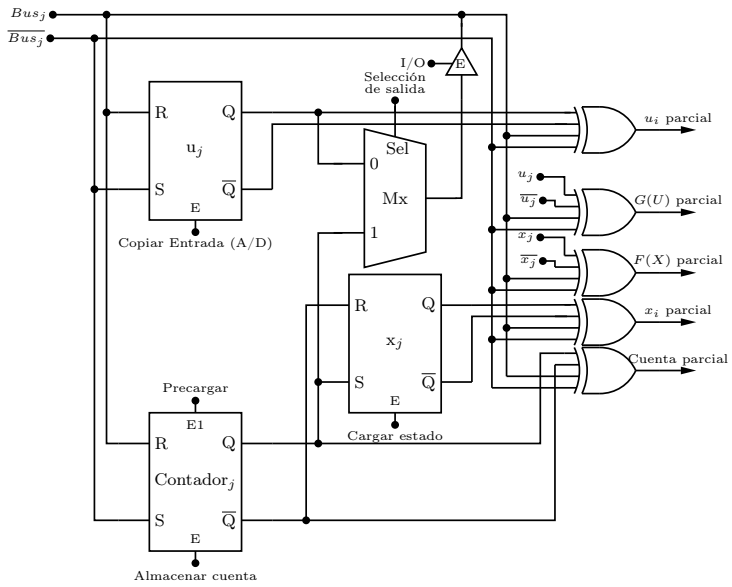
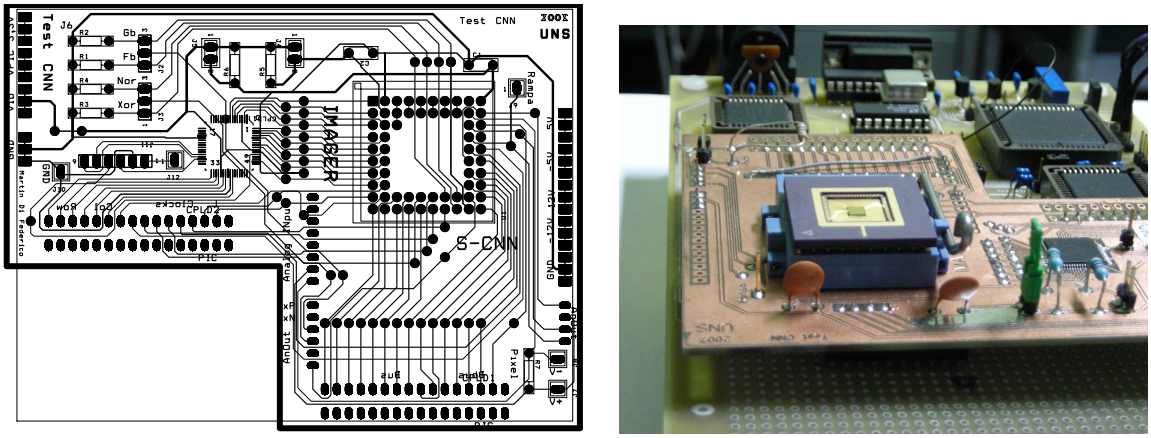


Figura 7.11: Esquemático de la parte de la celda que realiza las comparaciones con el BUS.

Para realizar las mediciones se fabricó una placa de testeo que se conecta a la placa de testeo genérica. La Fig. 7.12 muestra una fotografía de la placa de testeo montada sobre la placa genérica. Con esta placa se testearon las características básicas del circuito y se realizó el procesamiento de imágenes. Los ensayos se realizaron con el reloj funcionando a $10MHz$.

En el primer paso de la verificación se comprobó el funcionamiento individual



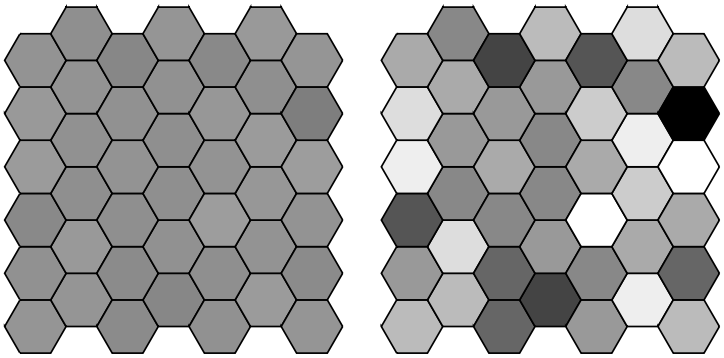
a) b)

Figura 7.12: a) Máscara de la placa fabricada; b) Fotografía del sistema completo de verificación.

de cada una de las partes. Se dispusieron las señales de control, las entradas y salidas, para habilitar parcialmente los diferentes circuitos internos. Por ejemplo, el funcionamiento de los fotodetectores se verificó polarizando los fotodiodos con una tensión inversa y midiendo la corriente que ingresaba al nodo V_{Diode} al variar la intensidad de la luz incidente (mientras el resto del integrado se mantenía en reposo y la señal de V_{Reset} se mantenía en alto). Para la verificación funcional del comparador, se ingresaron las entradas de comprobación en forma directa en las líneas V_{Sample} y V_{Ramp} y se verificó el momento de la retención del valor en u_j dentro de la celda unitaria (Fig. 7.11). Se repitió el ensayo ingresando las señales por las líneas V_{Reset} y V_{Sample}.

Verificación del fotodiodo: Para verificar la adquisición, se estableció una tensión fija en los fotodiodos y se realizó la conversión A/D. Dado que todos los pixeles poseen la misma tensión, idealmente es esperable obtener una imagen homogénea, pero en realidad lo que se obtiene es el patrón de ruido fijo ó "FPN"(fixed pattern noise).

En la Fig. 7.13 se puede ver una muestra de la imagen obtenida por el imager colocando una tensión fija en la entrada V_{Reset}; la Fig 7.13 a) muestra la conversión simultánea de todos los pixels mientras que la Fig. 7.13 b) muestra a máxima escala



a) b)

Figura 7.13: a)Imagen con V_{Reset} fijo en 2.5V; b)Ampliación a máxima escala del ruido FPN

de grises las diferencias entre los pixeles para mostrar el ruido de patrón fijo "FPN" del arreglo. Los valores más bajos son negro y los más altos son blanco.

Otro ensayo realizado fue la comprobación funcional de cada elemento de las celdas y del bloque de control de cada celda (Fig. 7.11), arbitrando las señales de control e inicializando los registros, de manera de forzar los cambios de estado y las operaciones particulares de cada lógica. Asi se almacenaron valores en los registros y se leyeron valores verificando el correcto funcionamiento de los registros de las celdas del comparador que genera la señal PWM, y de bloque FoG.

Verificación del procesamiento:

Finalizada la comprobación de los bloques individuales, se programó en la placa de verificación el algoritmo de control del integrado incluyendo los ciclos de programa y de evaluación, dejando accesible la programación S-CNN desde Matlab. La Fig. 7.14 muestra la ventada del programa que envía y recibe las señales del circuito.

A continuación se muestran algunas imágenes procesadas que con este circuito. La imagen que se utilizó para realizar el procesamiento es la imagen de patrón de ruido fijo, dado que en ese momento no se contaba con la óptica necesaria para hacer foco en el circuito. En el apéndice A se encuentra una biblioteca de funciones que los circuitos integrados con procesamiento S-CNN pueden realizar.

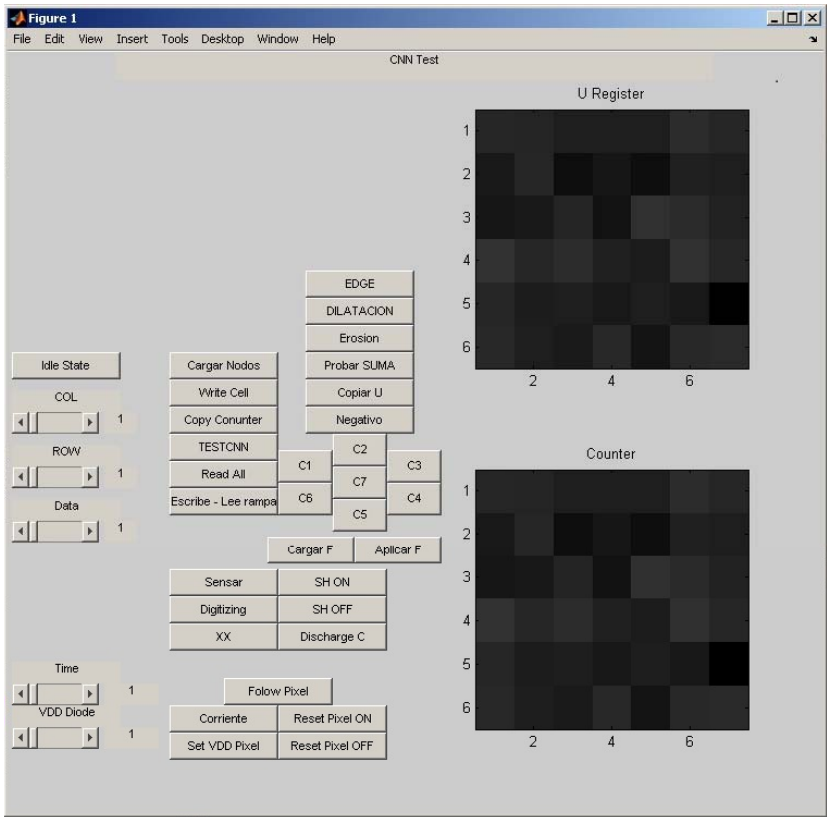


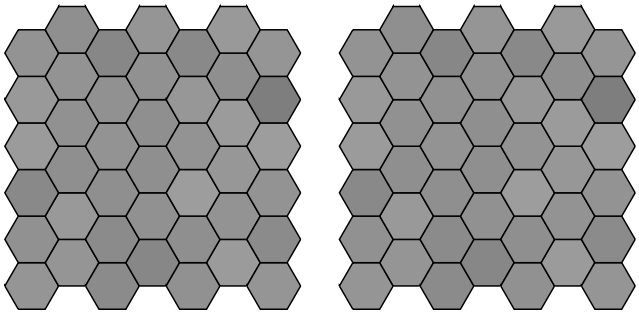
Figura 7.14: Interfaz de las mediciones con la Computadora.

- Copiar: La primer función en ser testeada es la función Copiar imagen. Esta función $G(U) = u_7$ copia el valor del registro en el contador. La Fig. Fig:testcopy muestra el valor del contador al finalizar la rampa de evaluación.

- Traslación: La Fig. 7.16 ilustra el funcionamiento del integrado cuando se ejecuta el programa de desplazamiento vertical.

- Erosión: En la Fig. 7.17 se ilustra el funcionamiento de un programa S-CNN de erosión de imagen (reducción de los bordes) donde se tomó como base para el procesamiento la imagen adquirida para demostrar el ruido FPN y se ejecutó el programa con condiciones de contorno de bordes blanco.

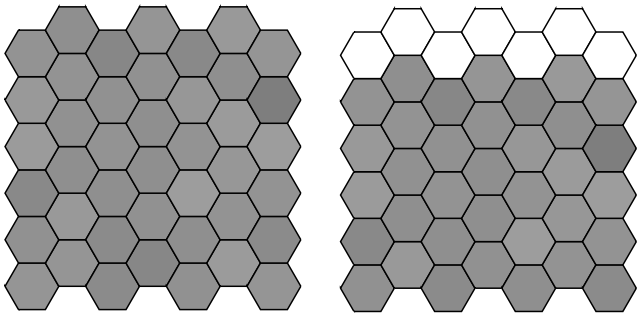
- Detección de bordes: La Fig. 7.18(b) es la imagen resultante luego de ejecutar el programa S-CNN de búsqueda de bordes a la Fig. 7.18(a).



a) b)

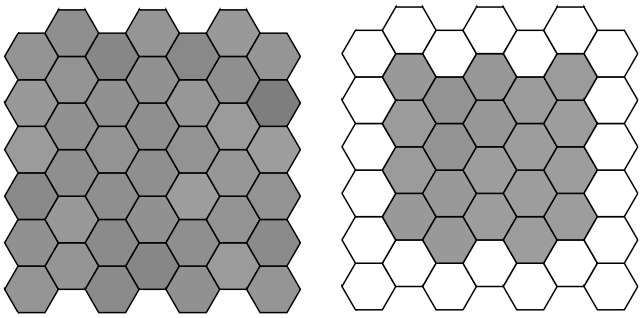
Figura 7.15: a)Imagen con V_{Reset} fijo en 2.5V; b)Resultado luego de aplicar el programa de copiado.

- Dilatación: Las Figs. 7.19 ilustra el funcionamiento del integrado cuando se ejecuta el programa de dilatación.



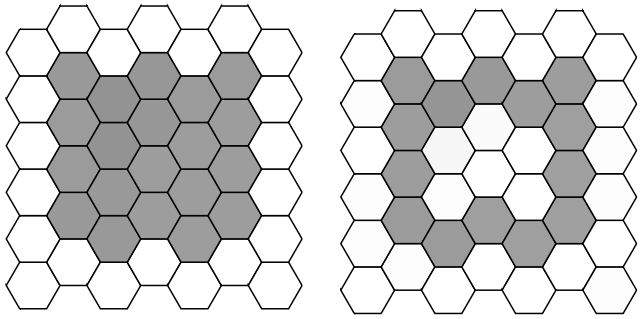
a) b)

Figura 7.16: a)Imagen original; b)Resultado luego de aplicar el programa de Traslación hacia abajo.



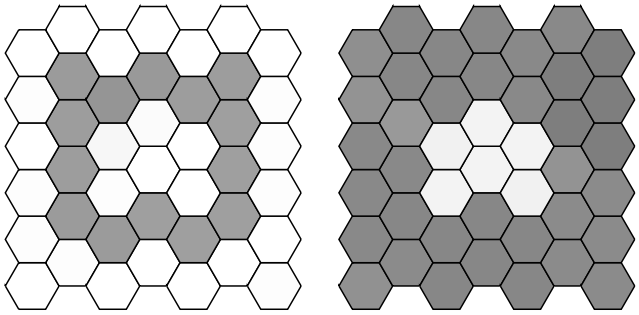
a) b)

Figura 7.17: a)Imagen con V_{Reset} fijo en 2.5V; b)Resultado luego de aplicar el programa de erosión.



a) b)

Figura 7.18: a)Imagen original; b)Resultado luego de aplicar el programa de búsqueda de bordes.



a) b)

Figura 7.19: a)Imagen original; b)Resultado luego de aplicar el programa de Dilatación.)

7.3.3. Celdas en tecnología TSMC 0,35

En esta sección se muestra un ejemplo de arquitectura realizada en una tecnología de 0, 35. En este caso se fabricaron y estudiaron dos tipos de arquitecturas digitales, una con una arquitectura CMOS, y otra utilizando lógica de pre carga. Lógicamente las dos celdas realizan las mismas operaciones, la arquitectura es la misma en las dos celdas, pero la forma de implementar la lógica en cada uno de los circuitos es diferente. Uno de los circuitos utiliza lógica complementaria o CMOS, donde el circuito posee dos redes de transistores, una red de transistores NMOS que se activa a la hora de colocar un çero.ª la salida, y una red con transistores PMOS que se activa cuando la salida es üno". Las dos redes son complementarias, lo que significa que nunca están las dos conectadas a la vez, logrando que la salida sea 0 o 1. En este caso, la red de P (pull-up, que escribe un 1) es tan grande como la red N (pull-down, que escribe un 0).El otro circuito, opera de una manera distinta, en vez de utilizar lógica complementaria, se utiliza un tipo de lógica, en la que solo se tiene una red de evaluación con transistores NMOS. Para sacar un dato valido, el proceso de evaluación debe constar de dos pasos. Primero se pre-cargan todas las salidas del circuito en 1 lógico, y luego se evalúa la salida con la red N. Si la salida debe ser un cero, la red N lo escribe (descarga el nodo de salida), y si debe ser un 1, la red N no se activa y queda el uno lógico que fue pre-cargado en el paso anterior. Este circuito al no poseer red P es más pequeño, pero necesita dos pasos para evaluar la salida. A continuación se describe la arquitectura de las dos celdas, y luego se detallan cada uno de los dos circuitos.

Descripción de la arquitectura de ambas celdas

Las celdas contienen dos registros llamados X y U. Tienen dos comparadores para generar las dos señales pwm y generar la dirección de memoria, Xf y Uf, que son la concatenación de las señales PWM propias y de los vecinos. Como cada celda tiene cuatro vecinos, el vector de entrada a la función es de cinco bits. Hay una sola memoria fuera de la celda, formada por un registro serie. Los datos se transmiten a las celdas de a uno a la vez, en 32 pasos de reloj, sincronizando los valores de la memoria con la dirección que se coloca en el bus de entrada. Esta rampa, donde se

evalúa el valor del vértice y se verifica el valor que se debe sumar, se llama Rampa Evaluación. Para realizar el proceso de evaluación de la función S-CNN (generar la señal PWL) se debe realizar otra rampa llamada Rampa de Programa que indique el vértice que debo integrar. Los valores que puede tomar de la memoria son de un bit, pueden ser 0 o 1, así que se implementa un integrador, mediante un contador de un bit, que en el caso de que la entrada sea uno evoluciona (cuenta) y si es cero no lo hace.

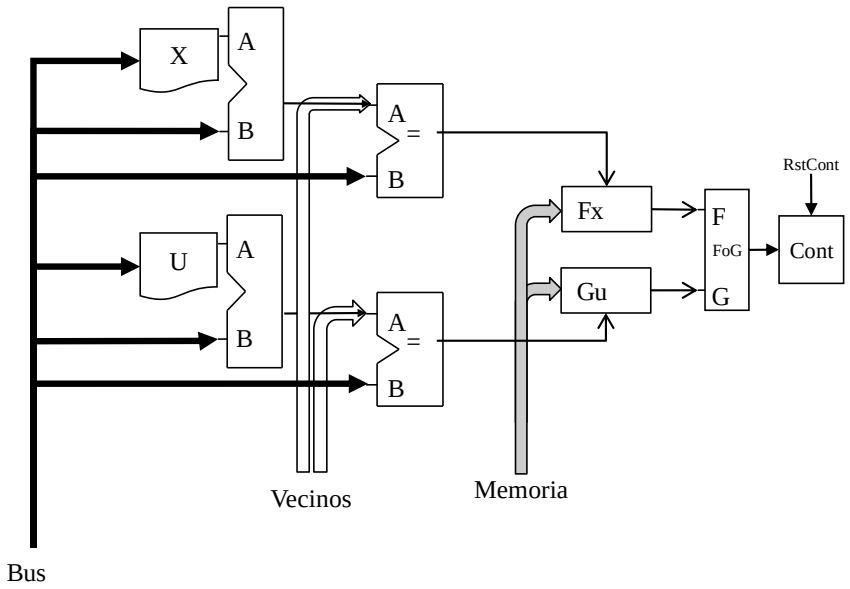


Figura 7.20: Bloques de la celda.

Ambas celdas se conectaron en paralelo, compartiendo gran parte de las señales de entrada. Con la señal Din/CMOS se selecciona la celda a usar, e inhibe la otra. Se utilizan veinte señales para realizar la comunicación con el circuito, las cuales se enumeran a continuación: I/O Bus(4..0) : Entrada y Salida del BUS Bus Dir : Dirección BUS (Entrada o Salida) BusComplement : Realizar el complemento del bus para evaluar con el circuito de lógica dinámica. ClkE, ClkP, ClkR, ClkNei, ClkFG, ClkO : Clocks PCb : Entrada de pre carga de nodos en lógica dinámica InSReg : Entrada del registro serie para cargar la memoria. Din/CMOS : Uso de celda Dinamica o Estatica. SelOut(2..0) : Selección de salida Out : Salida de nodo seleccionado

Celda estática

La celda estática usa lógica CMOS complementaria para realizar todas las operaciones necesarias para calcular la evolución de la celda. La entrada y los estados de la celda son almacenados en un FlipFlop que funciona con un reloj de fase simple con reset tipo D. El comparador usado para realizar la comparación del estado con la rampa digital, que genera las señales pwm, es un comparador modular. El comparador usado para comparar el vector de dirección con la dirección de la memoria transmitida es un comparador de igualdad, fabricado con compuertas lógicas xor. La función lógica que se aplica a los valores de las funciones de F y G se hace con el bloque FoG, que contiene tres compuertas básicas AND, OR, XOR, y con la señal de comando se utiliza un multiplexor para seleccionar la salida. Un contador modular es utilizado para integrar los valores de salida de la señal FoG.

Celda dinámica

De manera de lograr una implementación de área reducida se utilizó lógica dinámica de precarga y evaluación, lo cual permite deshacerse de la red de pull-up de transistores PMOS necesaria en realizaciones con lógica estática CMOS. La celda posee registros tipo SR con habilitación de reloj para el almacenamiento de todos los valores. El comparador usado para realizar la comparación del estado con la rampa digital y generar las señales pwm, se realiza precargando los nodos parUPWM y parXPWM y descargándolos con una media compuerta XOR, mostrada en la Fig. 7.22 implementada bit a bit.

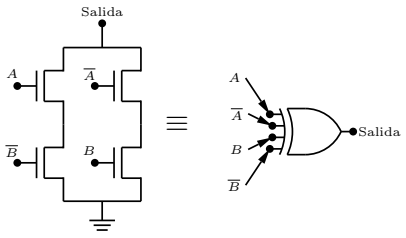


Figura 7.21: Esquemático del comparador realizado con una pseudo compuerta XOR.

El nodo de salida primero es pre cargado, luego la información es evaluada, y el

nodo es descargado si hay una diferencia entre las entradas. La Fig. 7.22 muestra el esquemático de la sección que muestra el nodo que se pre carga al comienzo de la evaluación, y los registrros SR que almacenan el valor de la señal pwm.

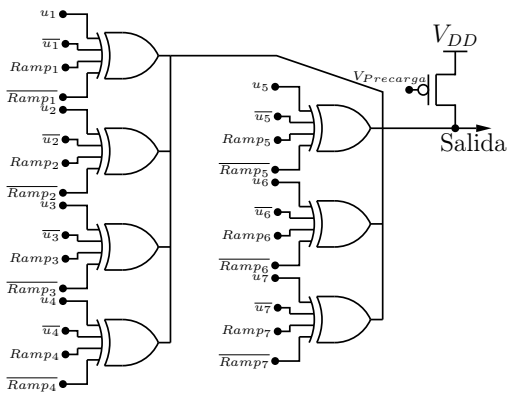
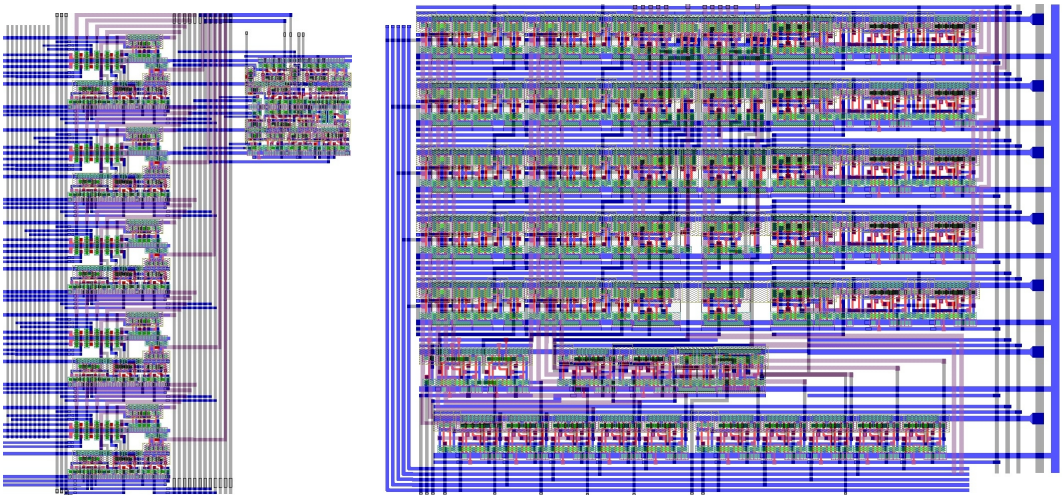


Figura 7.22: Nodo de precarga.

El comparador utilizado para la obtención del valor de memoria es igual que el comparador explicado anteriormente. La operación lógica entre los valores obtenidos de la memoria se realiza con el bloque FoG que computa la función lógica entre las señales G(uf) y F(xf) obtenidas de la memoria. El contador ha sido implementado indirectamente aprovechando la rampa de evaluación que es monótona y creciente. Al final de la rampa de programa, si el contador (almacenado en un registro) debe ser incrementado en uno, una bandera es establecida, y en la siguiente rampa de evaluación, un comparador de igualdad detecta que la rampa es igual al valor actual del contador. Luego, en el próximo paso, el valor de la rampa es almacenado, logrando así incrementar en uno el contador. Esto elimina la necesidad de un contador, y puede ser realizado de manera muy eficiente utilizando un comparador de igualdad. Se debe evitar que se comparen los valores en la rampa de programa (en vez de la rampa de evaluación), dado que las dos rampas se ingresan por la entrada BUS. Esta habilitación de comparación se hace por medio de una entrada específica.

La máscara de ambas celdas se muestra en la Fig. 7.23. La imagen de la izquierda es la celda dinámica, y la de la derecha la celda estática. El diseño de la celda dinámica es más complejo, tiene mayor cantidad de señales y es más lento en su aplicación,



a) b)

Figura 7.23: a) Máscara de la celda dinámica; b)Máscara de la celda CMOS

dado que se necesitan dos pasos para evaluar un valor. El tamaño de la celda dinámica es de aproximadamente 8,000μm^2, y el de la celda estática es de 18,0000μm^2. Como se puede ver, el tamaño de la celda dinámica es considerablemente más chico. Una fotografía del chip se muestra en la Fig. 7.24

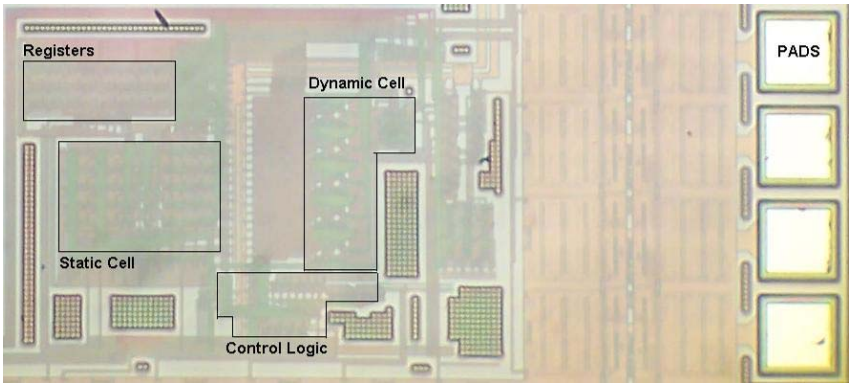


Figura 7.24: Fotografía del chip completo.

Descripción de la verificación.

Esta sección describe las mediciones realizadas y los resultados obtenidos de la verificación del circuito integrado explicado. Las mediciones se realizaron con el sistema de mediciones de chip genérico que fue explicado en la sección 7.2. Para realizar las mediciones de este chip, se fabricó una placa de testeo que se encastra en la placa de testeo genérica. La Fig. 7.26 muestra la máscara del circuito y una fotografía de la placa que se monta sobre el conector. Con esta placa se testearon todos los bloques del circuito. Los ensayos se realizaron con un reloj cuya velocidad es de $10MHz$; por lo tanto esta es la velocidad de operación del chip.

Para comunicarse con el circuito se creó una interfaz gráfica en MatLab® para enviar y recibir a información del circuito a verificar. La Fig. 7.14 muestra la ventada del programa que envía y recibe las señales del circuito.

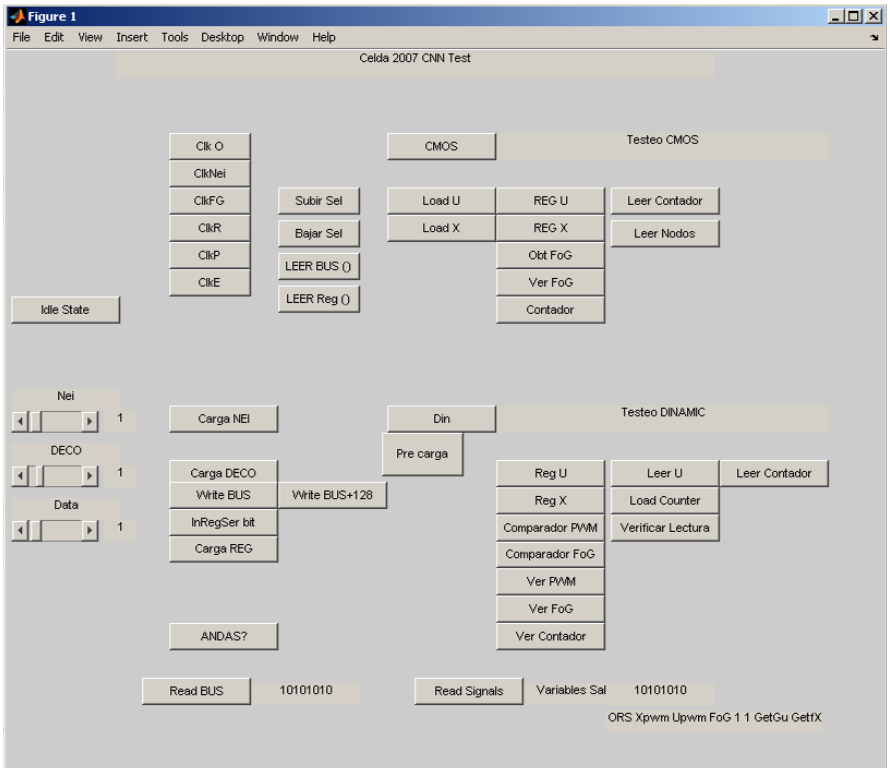
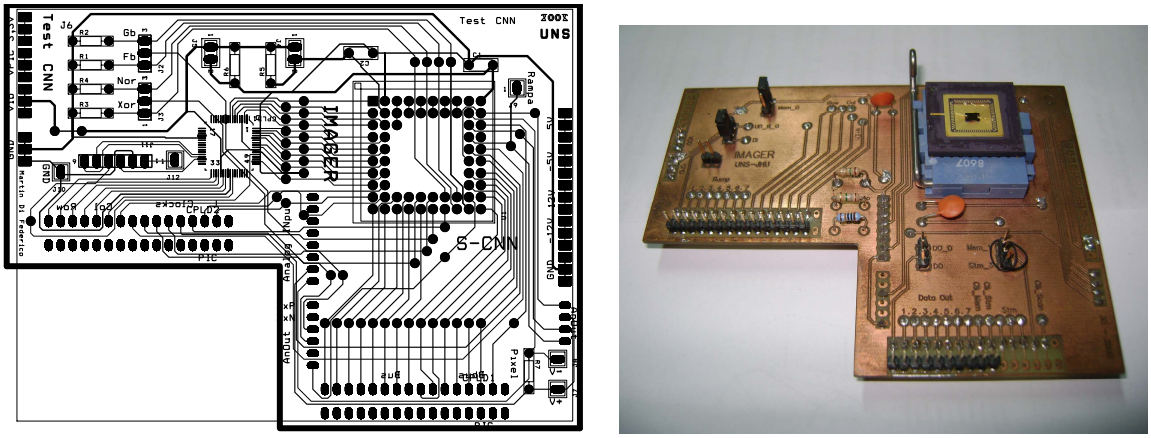


Figura 7.25: Interfaz de las mediciones con la Computadora.



a) b)

Figura 7.26: a) Máscara de la placa fabricada; b)Fotografia de la placa usada para las verificaciones

Como primer paso en la verificación se comprobó el funcionamiento de los registros de la celda. Se almaceno un valor en los registros y luego se leyó para cotejar el procedimiento de escritura y lectura. Se verifico también el funcionamiento de los registros series que almacenan la información de los vecinos y luego se procedió a la verificación de la procesamiento de la celda.

En la verificación de ambas celdas los datos de los vecinos se colocaron externamente a la celda. Dado que es una celda aislada se debe tener el valor de la señal pwm de los vecinos. Para el caso de las mediciones mostradas, el valor de los vecinos queda definido con un valor 5 para la función Uf y 10 para la función Xf:
Uf = 00101b (5d)
Xf = 01010b (10d)
Para realizar las En ambos casos la entrada tiene un valor 15, y el estado un valor de 20.
Xi = 10100 b (20d)
Ui = 01111 b (15d)

Verificación de la celda CMOS Los resultados de la verificación de la celda estática se muestran en la Fig. 7.27. Las señales UPWM y XPWM son las resultantes de comparar el valor de la entrada y el estado con la rampa digital del bus. Las señales GetFx y GetGu se utilizan para capturar el valor de la memoria $G(uf)$ y $F(xf)$ durante la rampa de evaluación. (La rampa de evaluación es el procedimiento por el cual toda la memoria es transmitida a las celdas). Estas señales deben valer $'1'$ cuando el bus (que indica la dirección del dato de memoria que se está enviando) es igual a la dirección del vértice de memoria que debe leer (indicado por la vecindad) Las señales XPWM y UPWM son generadas durante la rampa de programa. Durante la rampa de programa, las señales $GetGu$ y $GetFx$ son generadas pero no se amacena el valor de la memoria en los registros, y durante la rampa de evaluación, las señales XPWM y UPWN no son latcheadas. La señal FoG es la salida de la operación entre Fx y Gu. Este valor entra al contador indicando si debe o no contar. En este caso vale$'1'$ de manera que el contador está contando.

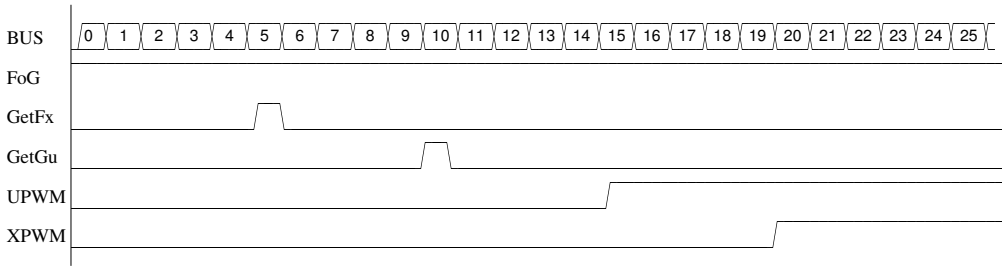


Figura 7.27: Resultados de la verificación de la celda CMOS.

Verificación de la celda dinámica Los resultados de la verificación de la celda dinámica se muestran en la Fig. 7.28. Todos los nodos con la petra p, son nodos pre cargados. la señal pFX es el resultado de la comparación entre el bus y los valores de los vecinos. Esta señal es usada para obtener los valores de la función de la memoria. Las señales $pUPWM$, $pXPWM$ se utilizan para almacenar en un latch las señales $UPWM$ y $XPWM$. La señal FoG es la salida de la operación entre Fx y Gu. Este valor entra al contador indicando si debe o no contar. En este caso vale $'0'$ de manera que el contador no está contando.

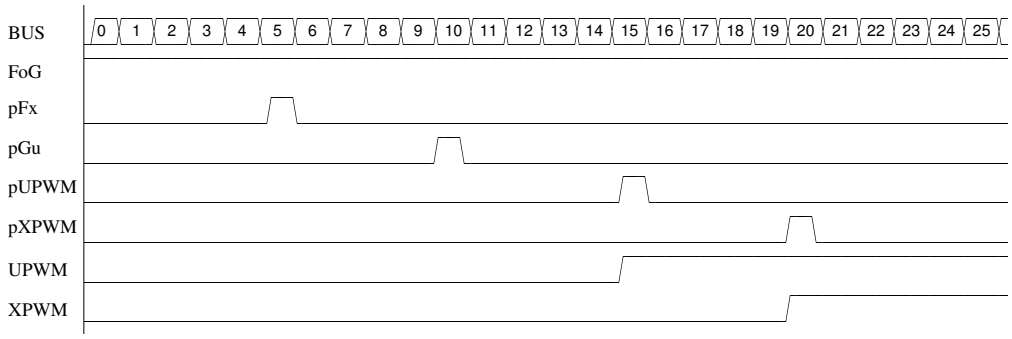
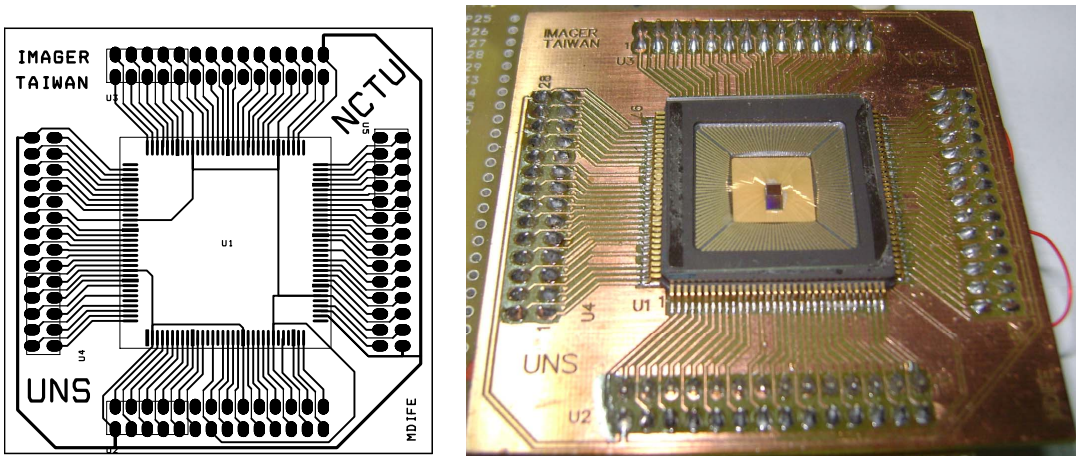


Figura 7.28: Resultados de la verificación de la celda dinámica.

7.3.4. Imager en tecnología 90nm

La arquitectura de este circuito integrado se encuentra descripto en el Cap. 5. Para testear este circuito se realizaron dos placas de testeo. Una para ser usada en la placa genérica, cuyas conexiones se realizan con cableado tipo "*wirewrap*" y otra conectada a la placa de desarrollo Memec con la FPGA de Xilinx.

La primera, que se conecta a la placa genérica, se muestra en la Fig. 7.29. Esta posee 32 señales de entrada y salida. Dado que el chip posee más de 60 señales, solo algunas partes del circuito fueron verificadas. Con esta placa se midió el consumo del circuito, a distintas frecuencias, se verificó el funcionamiento de las memorias de programa y se realizó el gráfico Shmoo, que indica las regiones donde el circuito anda.

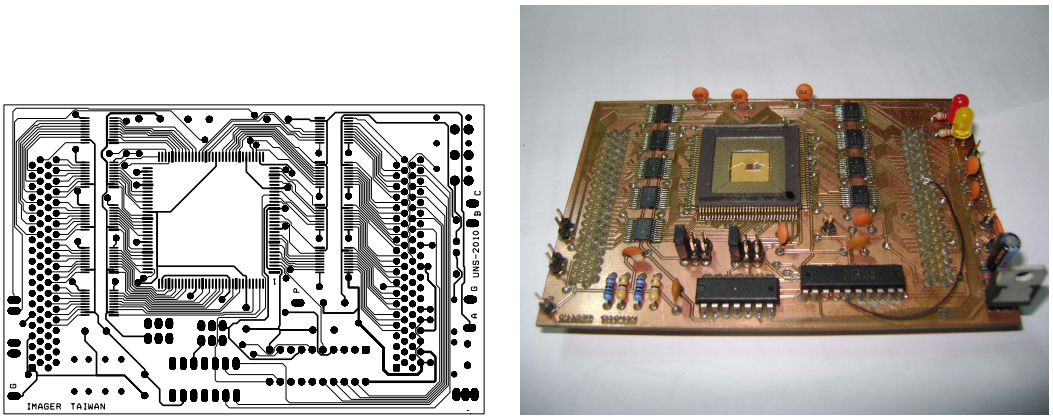


a) b)

Figura 7.29: a) Máscara de la placa de mediciones del chip de 90nm; b)Fotografia de la placa usada para las verificaciones en el chip de 90nm

Luego de realizar las mediciones de consumo y de funcionamiento básico se fabricó otra placa para realizar mediciones más complejas en las que más señales son necesarias. La placa mostrada en la Fig. **??** genera todas las señales necesarias para manejar todas las características del chip y también puede leer todas las señales que el chip envía. Con esta placa se verificó el funcionamiento de la memoria de programa, de los registros de todas las celdas, del sumador de filas, de los conversores

AD, se verificó la adquisición de imágenes y el procesamiento de imágenes utilizando varias configuraciones de vecinos y de operaciones entre funciones.



a) b)

Figura 7.30: a) Máscara de la placa de mediciones del chip de 90nm; b)Fotografia de la placa usada para las verificaciones en el chip de 90nm

Para comunicarse con el circuito se creó una interfaz gráfica en MatLab® para enviar y recibir a información del circuito a verificar. La Fig. 7.31 muestra la ventada del programa que envía y recibe las señales del circuito.

Descripción de la verificación.

A continuación se describe el procedimiento de las mediciones realizadas y se muestran algunos datos obtenidos.

Verificación: A continuación se detalla la verificacion de la Memoria de programa, de los registros de las celdas, se verifico la zona de trabajo en frecuencia y tensión, el consumo dinamico y estatico, mediciones del ADC y diferentes procesamientos de imagenes.

- Memoria:

  El siguiente paso en la comprobación fue la verificación de la memoria donde se almacenan las Funciones F y G. Esta memoria (un registro serie) fue verificada introduciendo datos, y verificando que la salida concuerde con la entrada. Se

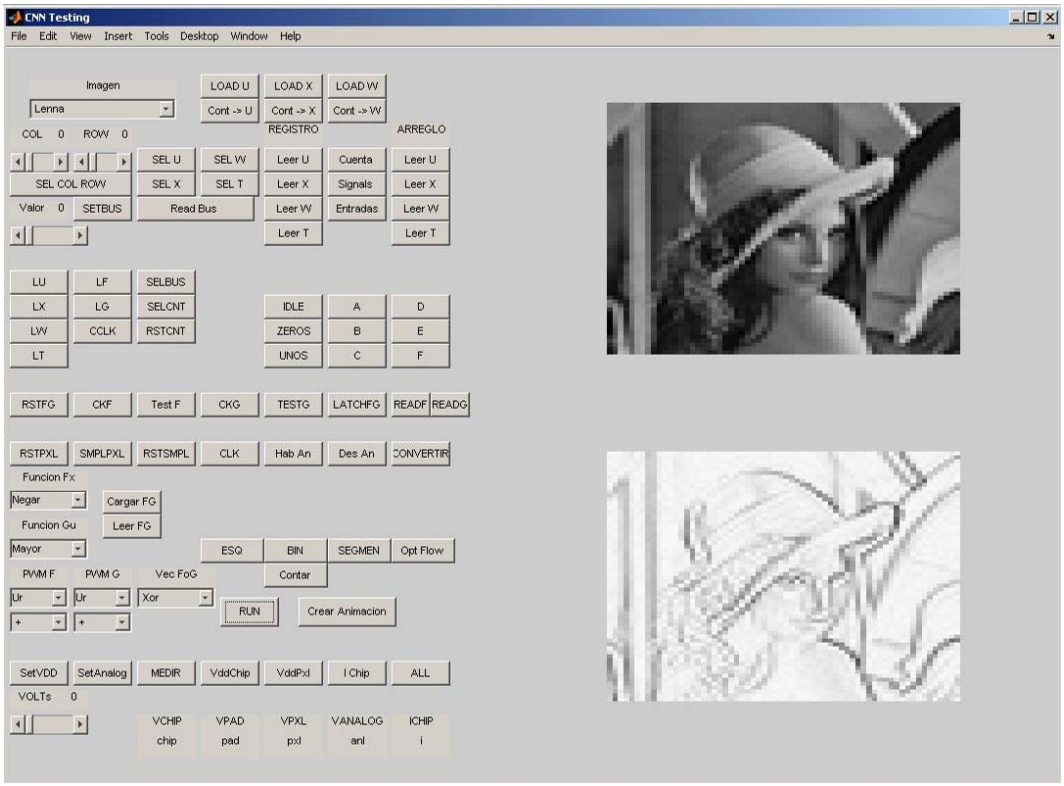


Figura 7.31: Interfaz con la computadora para realizar las mediciones del chip de 90nm.

utilizaron diferentes vectores de entrada, y la salida concordaba en todos los casos. Los datos fueron medidos con Matlab con el cual se verificó el correcto funcionamiento del bloque.

- Arreglo de celdas:

  Como primer paso en la verificación del arreglo se almacenó un valor constante en las celdas y luego se leyeron (manteniendo el valor del bus de entrada en el mismo valor) a una velocidad de 1Mhz. En este caso la verificación fue satisfactoria. Como segundo paso, se almacenó en los registros de las celdas el valor de la columna a la cual corresponde ($C_{ij} = j$) y luego se realizó la lectura de los registros. El resultado fue satisfactorio, y en cada una de las celdas el dato leído fue igual al dato escrito previamente. Luego se realizó la misma tarea almacenando en las celdas el valor de su fila correspondiente ($C_{ij} = j$) y

el resultado de la medición también resulto positiva.

- Medición de zona de trabajo:
  Sabiendo que en condiciones normales (1,2V a 1Mhz) el arreglo de celdas funciona, se realizó un barrido en frecuencia y en tensiones para ver cuáles son los márgenes de funcionamiento del circuito. Para realizar esta tarea, se escribió todo el arreglo con diferentes valores, y luego se leyó el arreglo de celdas completo. Si todas las celdas eran iguales el resultado de la medición se marca como positivo, y el circuito funciona correctamente, si en alguna de las celdas no concuerda el valor leído con el escrito previamente el resultado del test se marca como negativo, indicando que hubo algún error. Esta medición mostro que el circuito funciona a partir de los $0,7V$ para todo el rango de frecuencia medido.

- Medición de consumo :
  Medición de consumo estático: Una vez conectado todo el circuito en la placa se procedió a medir el consumo de corriente estática del circuito. Para medir esta corriente se utilizó una resistencia de 14 Ohm en serie con el circuito. Esta corriente era de $2mA$ para una tensión de alimentación de 1,2V.

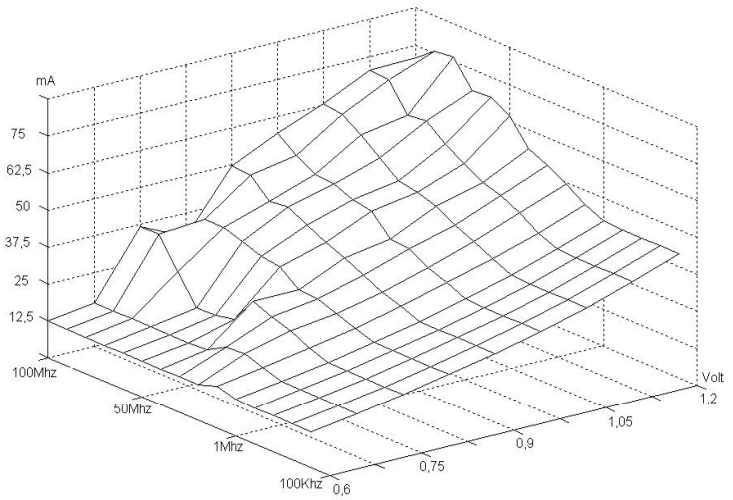


Figura 7.32: Consumo del circuito integrado medido en miliAmperes, en función de la frecuencia y la tensión de alimentación.

Medición de consumo dinámico: Se realizó un barrido en frecuencia y en tensiones para analizar el consumo del circuito integrado. Esta tarea se realizó leyendo valores de las celdas, dado que cada celda debe escribir en el bus de salida y es una de las acividades que más potencia consume. Los resultados de esta medición se muestran en la Fig. 7.32.

- Medición de los ADC:
  La Fig. 7.33 muestra las curvas transferencia de los conversores AD. El eje X indica el valor colocado directamente en el comprador, y el eje Y es el valor digital leído luego de la conversión. Cada uno de los graficaos muestra 3 valores, el valor medio, el mínimo y el máximo obtenido luego de la conversión AD de los 4096 conversores.

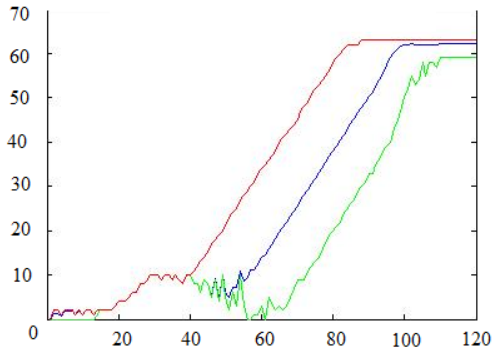


Figura 7.33: Curvas transferencia de los conversores: valores máximo, medio y mínimo. A/D.

Para realizar la conversión en cada uno de los pasos, se corrió una rampa analógica descendiente, (sincronizada con una digital de 63 a 0) teniendo en cuenta que al valor 63 le corresponden 450mV y al valor 127 le corresponden 913mV. Esta rampa (descendiente) se colocó en el pin VddPixel, con la señal de reset activa, de manera de simular la descarga del diodo. Esta tensión, luego de pasar por un seguidor, entra al comparador que realiza la conversión AD. La Fig. 7.34 muestra las señales del osciloscopio para realizar las conversiones.

La Fig.7.35 muestra la dispersión de los conversores. El eje X indica el valor

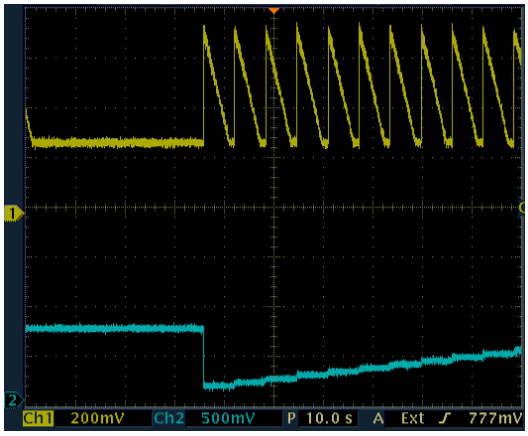


Figura 7.34: Imagen del osciloscopio de la conversión A/D.

colocado directamente en el comprador, el eje Y es el valor digital leído luego de la conversión y el eje Z indica la cantidad de comparadores con ese valor. Esta figura muestra los histogramas de los valores obtenidos luego de haber realizado las converiones AD.

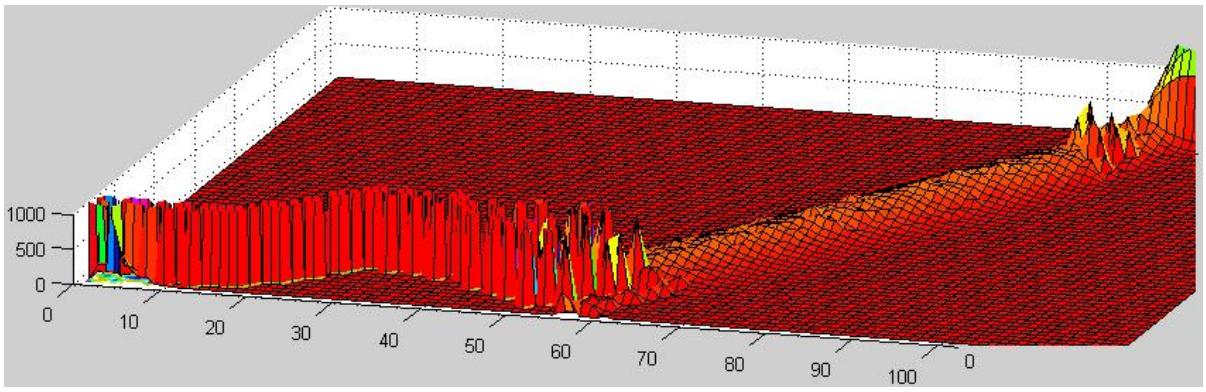


Figura 7.35: Curva de todos los conversores del arreglo.

La entrada del comparador se barrió de 0 a 900mV (0 a 120 en el conversor D/A), por eso el eje X tiene 120 valores, el eje Y indica cada uno de los 63 posibles valores que puede tomar la celda. La altura del grafico indica cuantos conversores tienen ese valor luego de la conversión. La Fig. 7.36 muestra la

imagen que se obtiene al colocar un valor de 450mV en la entrada del comparador.

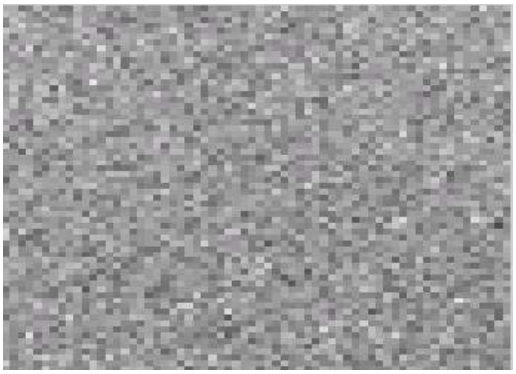
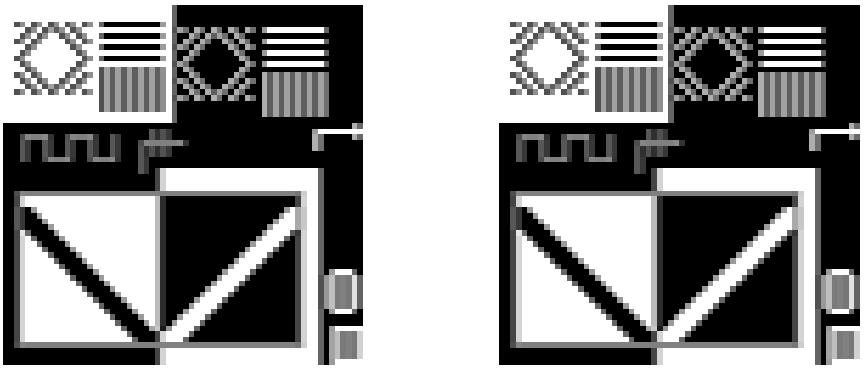


Figura 7.36: Imagen de patrón de ruido fijo o FPN.

- procesamiento de imágenes

  A continuación se muestran algunas imágenes procesadas que con este circuito. La imagen que se utilizó para realizar el procesamiento es la imagen patrón de ruido fijo, dado que en ese momento no se contaba con la óptica necesaria para hacer foco en el circuito. En el apéndice A se encuentra una biblioteca de funciones que los circuitos integrados con procesamiento S-CNN pueden realizar.

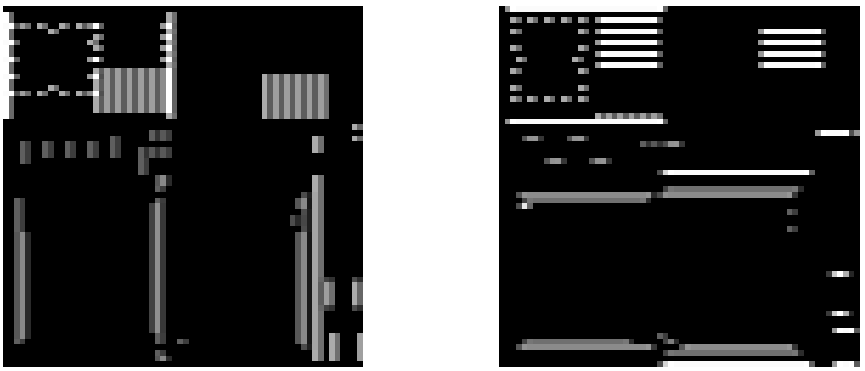
- Copiar: La primer función en ser testeada es la función Copiar imagen. Esta función $G(U) = u_7$ copia el valor del registro en el contador. La Fig. 7.37 muestra el valor del contador al finalizar la rampa de evaluación.



a) b)

Figura 7.37: a)Imagen almacenada en el registro U y copiada al registro X.

- Eliminación de líneas Verticales y Horizontales: La Fig. 7.38 ilustra el funcionamiento del integrado cuando se ejecuta el programa de reconocimiento de líneas verticales y líneas horizontales.



a) b)

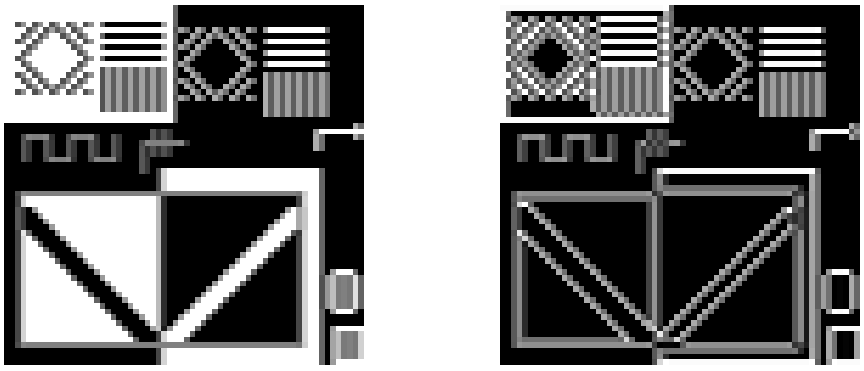
Figura 7.38: a)Muestra solo lineas Verticales; b)Muestra solo Lineas Horizontales.

- Búsqueda de Pixeles: En la Fig. 7.39 se muestra el procesamiento de dos diferentes procesos. En el primero se buscan pixeles aislados, es decir que no tengan ningún vecino en ninguno de sus cuatro lados. En la segunda imagen se muestran todos los pixeles que, estando dentro de una línea, están en sus extremos. La imagen Se almaceno en el registro U, y se ejecutó el programa con condiciones de contorno de bordes blanco.

a) b)

Figura 7.39: a)Busqueda de pixeles aislados; b)Busuqeda de finales de linea.

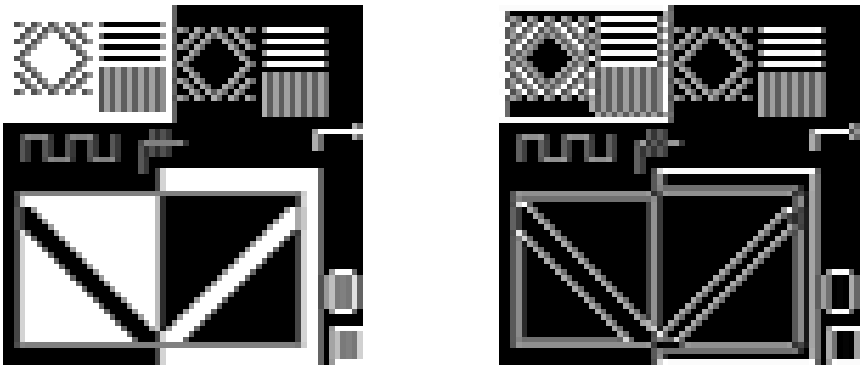
- Eliminación de relleno: La Fig. 7.40b) es la imagen resultante luego de ejecutar el programa S-CNN de eliminación de relleno a la Fig. 7.40a).



a) b)

Figura 7.40: a)Imagen original; b)Resultado de haber eliminado el relleno de las figuras.

- Detección de bordes: Las Figs. 7.41 muestra como se obtienen los bordes de la imagen realizando la operación entre dos funciones. Para realizar esta tarea se aplicó la Función XOR a la imagen original y a la imagen erosionada, quedando solo la diferencia entre las imágenes, que son los bordes. La Fig. 7.41(a) muestra la imagen original, almacenada en el registro U, y la Fig. 7.41(b) muestra el resultado de la operación.



a) b)

Figura 7.41: a)Imagen original; b) Resultado luego de aplicar el programa de Detección de bordes.)

Indice de terminos y Siglas

Terminos:

8051: Microprocesador

ADC: Conversion de analgico a digital

APS: Active píxel Sensor

CMOS: Tecnologia de fabricacion de circuitos integrados con transistores de efecto campo

CNN: Cellular non linear network

DIE: Material solido, cristalino de silicio, donde se fabrican los circuitos integrados

DFT: Design for Testeability

ESD: Electro Static Discharge

FPGA: Dispositivo con bloques de logica interconectables (Field Programable Gate Array)

GDS: Formato de archive, para intercambio de datos de mascaras de CI. (Graphic data system)

HMM: Modelo oculto de markov Hidden markov model o modelos markovianos ocultos

NMOS: Transistor de efecto campo, de sustrato tipo N

P&R: Place and route, Ubicación y conexionado

PAD:Dispositivo que se encuentra en los circuitos integrados, que permite la interconexión del chip con el encapsulado.

PCA: (Análisis de Componentes Principales)

PMOS: Transistor de sustrato tipo P

PRESICIÓN: Cantidad de bits con la que es convertida la imagen

PWL: Piece Wice Linear

PWM: Modulación de señal por ancho de pulso (pulse with modulation.)
RAMPA DE EVALUACION:Rampa utilizada parqa hacer la transmición de memoria donde se guardan los vertices
RAMPA DE PROGRAMA: Rampa digital para correr el algoritmo de procesamiento de la cnn simplicial.
Resolucion: Cantidad de pixels en la imagen definida por cantidad de columnas y cantidad de filas
SAD: Suma de las diferencias absolutas (Sum of Absolute Difference)
S-CNN: Simplicial Cellular non linear network
SIMD: (SingleInstruction multiple data) *Simplice*: En geometría, un símplice o n-símplice es el análogo en n dimensiones de un triángulo. Más exactamente, un símplice es la envoltura convexa de un conjunto de (n + 1) puntos independientes afines en un espacio euclídeo de dimensión n o mayor, es decir, el conjunto de puntos tal que ningún m-plano contiene más que (m + 1) de ellos. Por ejemplo, un 0-símplice es un punto; un 1-símplice un segmento de una línea; un 2-símplice un triángulo; un 3-símplice es un tetraedro; y un 4-símplice es un pentácoron (en cada caso, con su interior).
SoC: Sistem on Chip
SPI: Unidad de de transmicion y recepción de dastos serie sincronica
TSV: Interconección que pasa a través del silicio (Through silicon via)
UART: Unidad de transmición y recepción de datos asincrónica
UMC: Fabrica de circuitos integrados (www.umc.com)
USB: Universal serial BUS
VDD: Fuente de alimentación del circuito
VHDL: Lenguaje de descripcion de hardware (Very high scale integration Haredware Description Languaje)
Vt: Tensión umbral de los transisitores

Letras:
b = Cantidad de bits de la función a aplicar
C = Celda, celula o pixel.
Ci,j = Celda individual en la ubicación i,j

F = Filas

G = Columnas

K = Máximo valor de rampa

k = Valor de rampa

L = Lineas de funciones a enviar

m = Lugar de memoria

M = Tamaño de memoria

n = Cantidad de dimensiones / Vecinos

P = Numero de registros en la celda

q = Precisión de los registros de la celda

Q = Catidad de datos posibles en registro

Si,j = Vecindad de la celula i,j

U - Palabra digital: Valor del píxel de la conversión AD

X - Palabra digital: Valor del estado de la celda

W - Palabra digital: Valor del estado de la celda

T - Palabra digital: Valor del estado de la celda

Valores del chip CNN:

V(k): Dirección de la memoria a leer en el paso k de la rampa de programa

F(V(k)): Valor del vértice de la función F en el paso k de la rampa de programa

G(V(k)): Valor del vértice de la función G en el paso k de la rampa de programa

FPwm, GPwm: Señal PWM de X y de U (0 si es mayor que la rampa de programa, 1 si es menor que la rampa de programa)

VF = [FPwm(0) FPwm(1) FPwm(2) FPwm(3) FPwm(4) FPwm(5)]: Valor del vértice del símplice para evaluar la función F(Vf)

VG = [GPwm(0) GPwm(1) GPwm(2) GPwm(3) GPwm(4) GPwm(5)]: Valor del vértice del símplice para evaluar la función G(Vg)

G(Vg), F(Vf): Valor de la función en el vértice

Apéndice A

Biblioteca de programas S-CNN

En este apéndice se incluye una biblioteca de programas para un *imager* S-CNN con esfera de influencia de radio unitario (ver Fig. A.1) con una vecindad de 9 vecinos.

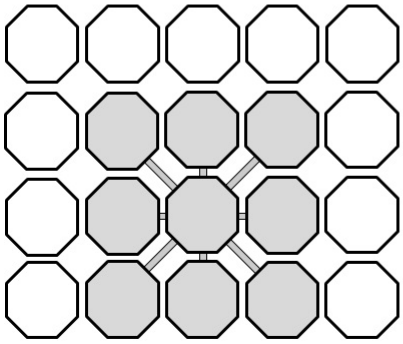


Figura A.1: Estructura de esfera de influencia de radio unitario.

Detección de Bordes

Función:	$G(U) = u_9\ F(X) = \overline{x_9}$
FoG	G(U)
Estado Inicial	No se utiliza
Entrada	Imagen
Condicion de borde	$x_{i*j*} = 0, u_{i*j*} = 0$
Desarrollo:	Si un pixel negro posee al menos un pixel blanco en su vecindad entonces pertenece al borde y queda negro.

Detección de rincones

Función:	$G(U) = u_9\ F(X) = \overline{x_9}$
FoG	G(U)
Estado Inicial	No se utiliza
Entrada	Imagen
Condicion de borde	$x_{i*j*} = 0, u_{i*j*} = 0$
Desarrollo:	Un pixel se mantiene negro si un fila y una columna de la vecindad del pixel contienen todos pixeles blancos.

Conversión a Blanco y Negro

Función:	$G(U) = u_9\ F(X) = \overline{x_9}$
FoG	G(U)
Estado Inicial	No se utiliza
Entrada	Imagen
Condicion de borde	$\mathrm{x}_{i*j*} = 0, u_{i*j*} = 0$
Desarrollo:	Para realizar la conversión a blanco y negro correspondiente al mapeo no lineal $[0, limite) \rightarrow 0$ y $[limite, 255] \rightarrow 255$ se congela la rampa de programa en el valor del *limite* y se ejecuta la función identidad $F(U) = u_9$.

Traslación horizontal

Función:	$G(U) = u_9\ F(X) = \overline{x_9}$
FoG	G(U)
Estado Inicial	No se utiliza
Entrada	Imagen
Condicion de borde	$\mathrm{x}_{i*j*} = 0, u_{i*j*} = 0$
Desarrollo:	

Traslación vertical

Estado inicial: No se utiliza

Entrada: Imagen

Condición de borde:

$$x_{i*j*} = 0, u_{i*j*} = 0$$

Función:

$$FoG = G(U)$$

$$\text{Up} \quad : \qquad G(U) = u_6$$
$$\text{Down} \quad : \qquad G(U) = u_2$$

Traslación vertical

Función:	$G(U) = u_9 \ F(X) = \overline{x_9}$
FoG	G(U)
Estado Inicial	No se utiliza
Entrada	Imagen
Condicion de borde	$\mathrm{x}_{i*j*} = 0, u_{i*j*} = 0$
Desarrollo:	

Traslación diagonal

Entrada: Imagen *Función:*	$G(U) = u_9 \ F(X) = \overline{x_9}$
FoG	G(U)
Estado Inicial	No se utiliza
Entrada	Imagen
Condicion de borde	$\mathrm{x}_{i*j*} = 0, u_{i*j*} = 0$
Desarrollo:	

Extracción de puntos aislados

Función:	$G(U) = u_9\ F(X) = \overline{x_9}$
FoG	G(U
Estado Inicial	No se utiliza uad
Entrada	
Condicion de borde	$x_{i*j*} = 0, u_{i*j*} = 0$
Desarrollo:	Si un pixel es negro y sus pixeles circundantes son blancos el punto se define como aislado y el resultado es un punto negro.

Remoción de puntos aislados

Función:	$G(U) = u_9\ F(X) = \overline{x_9}$
FoG	G(U)
Estado Inicial	No se utiliza
Entrada	Imagen
Condicion de borde	$x_{i*j*} = 0, u_{i*j*} = 0$
Desarrollo:	Si un punto es negro y todos sus vecinos son blancos el resultado es blanco.

Inversión de la imagen

Función:	$G(U) = u_9\ F(X) = \overline{x_9}$
FoG	G(U)
Estado Inicial	No se utiliza
Entrada	Imagen
Condicion de borde	$\mathrm{x}_{i*j*} = 0, u_{i*j*} = 0$
Desarrollo:	

Producto lógico "AND"

Función:	$G(U) = u_9\ F(X) = \overline{x_9}$
FoG	G(U)
Estado Inicial	Imagen 1
Entrada	Imagen 2
Condicion de borde	$\mathrm{x}_{i*j*} = 0, u_{i*j*} = 0$
Desarrollo:	

Suma lógica "OR"

Función:	$G(U) = u_9\ F(X) = \overline{x_9}$
FoG	G(U)+F(X)
Estado Inicial	Imagen 1
Entrada	Imagen 2
Condicion de borde	$\mathrm{x}_{i*j*} = 0, u_{i*j*} = 0$
Desarrollo:	

Suma lógica "XOR"

Función:	$G(U) = u_9\ F(X) = \overline{x_9}$
FoG	G(U) $\oplus F(X)$
Estado Inicial	Imagen 1
Entrada	Imagen 2
Condicion de borde	$\mathrm{x}_{i*j*} = 0, u_{i*j*} = 0$
Desarrollo:	

Detección de bordes con dirección de cambio preferencial

Función:	$G(U) = u_9\ F(X) = \overline{x_9}$
FoG	G(U)
Estado Inicial	No se utiliza
Entrada	Imagen
Condicion de borde	$\mathrm{x}_{i*j*} = 0, u_{i*j*} = 0$
Desarrollo:	

Erosión de una imagen

Función:	$G(U) = u_9\ F(X) = \overline{x_9}$
FoG	G(U)
Estado Inicial	No se utiliza
Entrada	Imagen
Condicion de borde	$\mathrm{x}_{i*j*} = 0, u_{i*j*} = 0$
Desarrollo:	Si el pixel es negro y cualquier vecino es blanco entonces el pixel pasa a blanco.

Dilatación de una imagen

Función:	$G(U) = u_9\ F(X) = \overline{x_9}$
FoG	G(U)
Estado Inicial	No se utiliza
Entrada	Imagen
Condicion de borde	$\mathrm{x}_{i*j*} = 0, u_{i*j*} = 0$
Desarrollo:	Si el punto es blanco y algún vecino es negro entonces el punto de convierte en negro

Proyección de sombra

Función:	$G(U) = u_9\ F(X) = \overline{x_9}$
FoG	F(X)
Estado Inicial	Imagen
Entrada	Imagen
Condicion de borde	$\mathrm{x}_{i*j*} = 0, u_{i*j*} = 0$
Desarrollo:	si un pixel es negro o su vecino contiguo en la dirección a proyectar es negro el resultado es negro. Para asegurar la proyección del total de la imagen se debe ejecutar un número "n" de veces esta operación. El numero de veces a repetir la operación está dterminado por la cantidad de pixeles lineales que posee la imagen en la dirección a proyectar.

Detección de líneas diagonales

Función:	$G(U) = u_9\ F(X) = \overline{x_9}$
FoG	G(U)
Estado Inicial	No se utiliza
Entrada	Imagen
Condicion de borde	$\mathrm{x}_{i*j*} = 0, u_{i*j*} = 0$
Desarrollo:	Si el pixel es negro y sus vecinos en la diagonal son negros el pixel continua siendo negro; caso contrario pasa a blanco.

Conectividad global

Función:	$G(U) = u_9\ F(X) = \overline{x_9}$
FoG	G(U)
Estado Inicial	Un pixel de la Imagen
Entrada	Imagen
Condicion de borde	$\mathrm{x}_{i*j*} = 0, u_{i*j*} = 0$
Desarrollo:	Cargar una imagen en el estado que posea un punto negro coincidente con el objeto a verificar. Luego de esto se hace "crecer" la imagen almacenada en el estado y se intersecta con la imagen original n veces. Al final se tendrá la misma imagen que la entrada si se cumple la condición de conectividad global del objeto.

llenado de agujeros

Función:	$G(U) = u_9\ F(X) = \overline{x_9}$
FoG	F(X)+G(U)
Estado Inicial	Imagen negra
Entrada	Imagen
Condicion de borde	$x_{i*j*} = 0, u_{i*j*} = 0$
Desarrollo:	Cargar una imagen negra en el estado. Erosionarla y unirla con la imagen a llenar en forma reiterada.

Extracción de contornos

Función:	$G(U) = u_9\ F(X) = \overline{x_9}$
FoG	G(U)
Estado Inicial	No se utiliza
Entrada	Imagen
Condicion de borde	$x_{i*j*} = 0, u_{i*j*} = 0$
Desarrollo:	Si el pixel es negro y contiene dos pixeles consecutivos blancos en su vecindad entonces el pixel pertenece a un borde y se mantiene negro. Esta condición puede restringirse a tres blancos consecutivos si se quiere aumentar la selectividad.

Detección de gradientes

Función:	$G(U) = u_9\ F(X) = \overline{x_9}$
FoG	G(U)
Estado Inicial	No se utiliza
Entrada	Imagen
Condicion de borde	$x_{i*j*} = 0, u_{i*j*} = 0$
Desarrollo:	Menos de cuatro pixeles negros en la esfera de influencia o más de seis negros en la esfera de influencia implican un pixel en blanco, el pixel se fija en negro para los demás casos.

Calculo momentos

Función:	$G(U) = u_9\ F(X) = x_9 Mscara$
FoG	G(U)
Estado Inicial	Imagen con valor de peso para el calculo de momento
Entrada	Imagen
Condicion de borde	$x_{i*j*} = 0, u_{i*j*} = 0$
Desarrollo:	Se realiza la copia de la Imagen cargada enmascarandola con la entrada. Luego se suman los valores de todos los registros y se lo divide el area. Con este algoritmo se puede calcular el centro de masa de un objeto, la desviación, o la curtosis.

Ubicación centro de masa

Función:	$G(U) = u_9\ F(X) = x_9 Mscara$
FoG	G(U)
Estado Inicial	Imagen donde cada celda contiene el valor de su fila, o columna
Entrada	Imagen
Condicion de borde	$\mathrm{x}_{i*j*} = 0, u_{i*j*} = 0$
Desarrollo:	Se realiza la copia de la imagen cargada enmascarandola con la entrada. Primero utilizando la imagen de las columnas, se suman todos los valores de la imagen resultante, se la divide por el area de la entrada, y asi se obtiene la coordenada X (columna) del centro de masa de la entrada. Luego se realiza la misma tarea con la imagen de las filas, se suman todos los valores y se divide por el area para obtener la coordenada Y (Fila).

Bibliografía

[1] V. Brea, D. Vilarino, A. Paasio, and D. Cabello, "Design of the processing core of a mixed-signal cmos dtcnn chip for pixel-level snakes," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 51, pp. 997 – 1013, May 2004.

[2] G. L. Cembrano, A. Rodriguez-Vazquez, R. C. Galan, F. Jimenez-Garrido, S. Espejo, and R. Dominguez-Castro, "A 1000 fps at 128×128 vision processor with 8-bit digitized I/O," *IEEE Journal of Solid-State Circuits*, vol. 39, pp. 1044–1055, 2004.

[3] C.-C. Cheng, C.-H. Lin, C.-T. Li, S. Chang, and L.-G. Chen, "ivisual: An intelligent visual sensor soc with 2790fps cmos image sensor and 205gops/w vision processor," in *Design Automation Conference, 2008. DAC 2008. 45th ACM/IEEE*, 2008, pp. 90 –95.

[4] C.-C. Cheng, C.-H. Lin, C.-T. Li, and L.-G. Chen, "ivisual: An intelligent visual sensor soc with 2790 fps cmos image sensor and 205 gops/w vision processor," *Solid-State Circuits, IEEE Journal of*, vol. 44, pp. 127 –135, 2009.

[5] M. Chien and E. Kuh, "Solving nonlinear resistive networks using piecewise-linear analysis and simplicial subdivision," *IEEE Transaction on Circuits Systems I*, vol. CAS-24, pp. 305–317, June 1977.

[6] L. O. Chua, "CNN: A vision of complexity," *International Journal of Bifurcation and Chaos*, vol. 7, pp. 2219–2425, October 1997, Special issue on Visions of Nonlinear Science in the 21st Century.

[7] L. O. Chua, T. Rosca, T. Kozek, and A. Zarandy, "CNN universal chips crank up the computing power," *IEEE Circuits Devices Magazine*, pp. 18–28, July 1996.

[8] L. O. Chua and L. Yang, "Cellular neural networks: Applications," *IEEE Transaction on Circuits Systems I*, vol. CAS-35, pp. 1273–1290, October 1988.

[9] L. O. Chua and L. Yang, "Cellular neural networks: Theory," *IEEE Transaction on Circuits Systems I*, vol. CAS-35, pp. 1257–1272, October 1988.

[10] L. O. Chua and T. Roska, *Cellular neural networks and visual computing : foundation and applications.* Cambridge, UK ; New York, NY: Cambridge University Press, 2005.

[11] L. Chua and T. Roska, "The cnn paradigm," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, vol. 40, pp. 147 –156, mar 1993.

[12] K. Crounse and L. Chua, "Methods for image processing and pattern formation in cellular neural networks: a tutorial," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, vol. 42, pp. 583 –601, oct 1995.

[13] J. Cruz and L. O. Chua, "A 16×16 cellular neural network universal chip: The first complete single-dynamic computer array with distributed memory and with gray-scale input-output," *Analog Integrated Circuits and Signal Processing Journal*, vol. 15(3), pp. 227–238, 1998.

[14] M. Di Federico, P. Mandolesi, P. Julian, and A. Andreou, "Experimental results of simplicial cnn digital pixel processor," *Electronics Letters*, vol. 44, pp. 27 –29, 3 2008.

[15] M. Di Federico, P. Mandolesi, P. Julian, and A. Andreou, "Experimental results of simplicial cnn digital pixel processor," *Electronics Letters*, vol. 44, pp. 27 –29, 3 2008.

[16] R. Dominguez-Castro, S. Espejo, A. Rodriguez-Vazquez, R. A. Carmona, A. Zarandy, P. Szolgay, T. Sziranyi, and T. Roska, "A 0.8-μm CMOS

two-dimensional programmable mixed-signal focal-plane array processor with on-chip binary imaging and instructions storage," *IEEE Journal of Solid-State Circuits*, vol. 32, pp. 1013–1026, 1997.

[17] R. Dominguez-Castro, S. Espejo, A. Rodriguez-Vazquez, R. Carmona, P. Foldesy, A. Zarandy, P. Szolgay, T. Sziranyi, and T. Roska, "A 0.8- mu;m cmos two-dimensional programmable mixed-signal focal-plane array processor with on-chip binary imaging and instructions storage," *Solid-State Circuits, IEEE Journal of*, vol. 32, pp. 1013 –1026, jul 1997.

[18] C. Dominguez-Matas, R. Carmona-Galan, F. Sanchez-Fernandez, and A. Rodriguez-Vazquez, "A focal-plane image processor for low power adaptive capture and analysis of the visual stimulus," in *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, May 2007, pp. 2690 –2693.

[19] J. Dubois, D. Ginhac, M. Paindavoine, and B. Heyrman, "A 10 000 fps cmos sensor with massively parallel image processing," *Solid-State Circuits, IEEE Journal of*, vol. 43, pp. 706 –717, 2008.

[20] P. Dudek, "Implementation of simd vision chip with 128 times;128 array of analogue processing elements," in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, May 2005, pp. 5806 – 5809 Vol. 6.

[21] P. Dudek and S. Carey, "General-purpose 128 times;128 simd processor array with integrated image sensor," *Electronics Letters*, vol. 42, pp. 678 – 679, 2006.

[22] P. Dudek and P. Hicks, "A general-purpose processor-per-pixel analog simd vision chip," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 52, pp. 13 – 20, jan. 2005.

[23] P. Dudek, A. Lopich, and V. Gruev, "A pixel-parallel cellular processor array in a stacked three-layer 3d silicon-on-insulator technology," in *Circuit Theory and Design, 2009. ECCTD 2009. European Conference on*, aug. 2009, pp. 193 –196.

[24] P. Dudek, S. Szczepanski, and J. Hatfield, "A high-resolution cmos time-to-digital converter utilizing a vernier delay line," *Solid-State Circuits, IEEE Journal of*, vol. 35, pp. 240 –247, feb 2000.

[25] D. Fey, L. Hoppe, and A. Loos, "Reconfigurable on-chip SIMD processor architectures for intelligent CMOS camera chips," in *Parallel Computing in Electrical Engineering, 2004. PARELEC 2004. International Conference on*, 2004, pp. 251–255.

[26] P. Foldesy, R. Carmona-Galan, A. Zarandy, C. Rekeczky, A. Rodriguez-Vazquez, and T. Roska, "Digital processor array implementation aspects of a 3d multi-layer vision architecture," in *Cellular Nanoscale Networks and Their Applications (CNNA), 2010 12th International Workshop on*, feb. 2010, pp. 1 –4.

[27] P. Foldesy, A. Zarandy, C. Rekeczky, and T. Roska, "3d integrated scalable focal-plane processor array," in *Circuit Theory and Design, 2007. ECCTD 2007. 18th European Conference on*, aug. 2007, pp. 954 –957.

[28] P. Foldesy, A. Zarandy, C. Rekeczky, and T. Roska, "High performance processor array for image processing," in *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, May 2007, pp. 1177 – 1180.

[29] Y. Joo, J. Park, M. Thomas, K. Chung, M. Brooke, N. Jokerst, and D. Wills, "Smart cmos focal plane arrays: a si cmos detector array and sigma-delta analog-to-digital converter imaging system," *Selected Topics in Quantum Electronics, IEEE Journal of*, vol. 5, pp. 296 –305, 1999.

[30] P. Julián, A. Desages, and B. D'Amico, "Orthonormal high level canonical PWL functions with applications to model reduction," *IEEE Transaction on Circuits Systems I*, vol. 47, pp. 702–712, May 2000.

[31] P. Julián, R. Dogaru, and L. O. Chua, "A piecewise-linear simplicial coupling cell for CNN gray-level image processing," *IEEE Transaction on Circuits Systems I*, vol. 49, pp. 904–913, July 2002.

[32] P. Kinget and M. Steyaert, "A programmable analog cellular neural network cmos chip for high speed image processing," *Solid-State Circuits, IEEE Journal of*, vol. 30, pp. 235 –243, mar 1995.

[33] J. Knickerbocker, P. Andry, B. Dang, R. Horton, C. Patel, R. Polastre, K. Sakuma, E. Sprogis, C. Tsang, B. Webb, and S. Wright, "3d silicon integration," in *Electronic Components and Technology Conference, 2008. ECTC 2008. 58th*, may 2008, pp. 538 –543.

[34] H. Kurino, T. Matsumoto, K.-H. Yu, N. Miyakawa, H. Itani, H. Tsukamoto, and M. Koyanagi, "Three-dimensional integration technology for real time micro-vision system," in *Innovative Systems in Silicon, 1997. Proceedings., Second Annual IEEE International Conference on*, oct 1997, pp. 203 –212.

[35] M. Laiho, J. Poikonen, P. Virta, and A. Paasio, "Live demonstration: Mipa4k: A 64 x00d7;64 cell mixed-mode image processor array," in *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*, May 2009, pp. 1931 –1931.

[36] Z. Lin, M. Hoffman, N. Schemm, W. Leon-Salas, and S. Balkir, "A cmos image sensor for multi-level focal plane image decomposition," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 55, pp. 2561 –2572, 2008.

[37] G. Linan, R. Dominguez-Castro, S. Espejo, and A. Rodriguez-Vazquez, "ACE16K: An advanced focal-plane analog programmable array processor," in *Proceedings of the 27th European Solid-State Circuits Conference, ESSCIRC 2001.*, 2001, pp. 201–204.

[38] G. Linan, A. Rodriguez-Vazquez, S. Espejo, and R. Dominguez-Castro, "ACE16K: a 128×128 focal plane analog processor with digital I/O," in *Proceedings of the 2002 7th IEEE International Workshop on Cellular Neural Networks and Their Applications (CNNA 2002).*, 2002, pp. 132–139.

[39] D. Lopez Vilarino, P. Dudek, and D. Cabello Ferrer, "Focal-plane moving object segmentation for realtime video surveillance," in *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, May 2008, pp. 1600 –1603.

[40] A. Lopich and P. Dudek, "Aspa: Focal plane digital processor array with asynchronous processing capabilities," in *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, May 2008, pp. 1592 –1595.

[41] P. Mandolesi, "Cámara cmos programable con procesamiento paralelo sobre el plano focal," *Tesis Doctoral*, 2007.

[42] W. Miao, Q. Lin, W. Zhang, and N.-J. Wu, "A programmable simd vision chip for real-time vision applications," *Solid-State Circuits, IEEE Journal of*, vol. 43, pp. 1470 –1479, 2008.

[43] P. R. E. Philip Garrou (Editor), Christopher Bower (Editor), *Handbook of 3D Integration: Technology and Applications of 3D Integrated Circuits*, volume 1 ed. Wiley, 2008.

[44] J. Poikonen, M. Laiho, and A. Paasio, "Mipa4k: A 64 x00d7;64 cell mixed-mode image processor array," in *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*, May 2009, pp. 1927 –1930.

[45] A. Rodriguez-Vazquez, G. Linan-Cembrano, L. Carranza, E. Roca-Moreno, R. Carmona-Galan, F. Jimenez-Garrido, R. Dominguez-Castro, and S. E. Meana, "ACE16K: the third generation of mixed-signal SIMD-CNN ACE chips toward VSoCs," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 51, pp. 851–863, 2004.

[46] A. Rodriguez-Vazquez, G. Linan-Cembrano, L. Carranza, E. Roca-Moreno, R. Carmona-Galan, F. Jimenez-Garrido, R. Dominguez-Castro, and S. E. Meana, "ACE16K: the third generation of mixed-signal SIMD-CNN ACE chips toward VSoCs," *Regular Papers, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 51, pp. 851–863, 2004.

[47] T. Roska and L. O. Chua, "The CNN universal machine: An analogic array computer," *IEEE Transaction on Circuits Systems I*, vol. CAS-40, pp. 289–291, 1993.

[48] T. Roska and L. Chua, "The cnn universal machine: an analogic array computer," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 40, pp. 163 –173, mar 1993.

[49] R. Serrano-Gotarredona, T. Serrano-Gotarredona, A. Acosta-Jimenez, C. Serrano-Gotarredona, J. Perez-Carrasco, B. Linares-Barranco, A. Linares-Barranco, G. Jimenez-Moreno, and A. Civit-Ballcels, "On real-time aer 2-d convolutions hardware for neuromorphic spike-based cortical processing," *Neural Networks, IEEE Transactions on*, vol. 19, pp. 1196 –1219, 2008.

[50] R. Takami, K. Shimonomura, S. Kameda, and T. Yagi, "An image pre-processing system employing neuromorphic 100 times; 100 pixel silicon retina [robot vision applications]," in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, May 2005, pp. 2771 – 2774 Vol. 3.

[51] T. Toi, "Color signal processing technique for single-chip CCD cameras that employ cpus with SIMD instruction sets," *Consumer Electronics, IEEE Transactions on*, vol. 46, pp. 291–294, 2000.

[52] A. Harton, M. Ahmed, A. Beuhler, F. Castro, L. Dawson, B. Herold, G. Kujawa, K. Lee, R. Mareachen, and T. Scaminaci, "High dynamic range CMOS image sensor with pixel level ADC and in situ image enhancement," in Sensors and Camera Systems for Scientific and Industrial Applications VI. Proc. SPIE, Mar. 2005, vol. 5677, pp. 67–77.

[53] A. J. Lipton et al., "The intelligent vision sensor: Turning video into information," in Proc. 2007 IEEE Int. Conf. Video and Signal Based Surveillance, 2007, pp. 63–68. C.-C. Cheng, C.-H. Lin, C.-T. Li, S. Chang, C.-J. Hsu, and L.-G. Chen, "iVisual: An intelligent visual sensor SoC with 2790 fps CMOS

image sensor and 205 GOPS/W vision processor," in IEEE ISSCC Dig. Tech. Papers, 2008, pp. 306–307.

[54] D. Parkinson, "The distributed array processor (DAP)," in Massively Parallel Computing With the DAP. Cambridge, MA: MIT Press, 1990.

[55] M. E. Ruaro, P. Bonifazi, V. Torre, "Toward the neurocomputer: image Processing and pattern recognition with neuronal cultures", IEEE Transactions on Biomedical Engineering, Mar 2005, vol 52, No 3, pp. 371 – 383

[56] E. R. Fossum, "CMOS image sensors: Electronic camera-on-a-chip," IEEE Trans. Electron Devices, vol. 44, no. 10, pp. 1689–1698, Oct. 1997.

[57] P. M. Flanders, D. J. Hunt, S. F. Reddaway, and D. Parkinson, "Efficient high speed computing with the distributed array processor," in Massively Parallel Computing With the DAP. Cambridge, MA: MIT Press, 1990.

[58] C. Garcia and M. Delakis, "Convolutional face finder: A neural architecture for fast and robust face detection," IEEE Trans. Pattern Anal. Machine Intell., vol. 26, no. 11, pp. 1408–1423, Nov. 2004.

[59] P. Robertson, R. Laddaga, and M. Van Kleek, "Virtual mouse vision based interface," in Proc. 9th Int. Conf. Intelligent User Interfaces, 2004, pp. 177–183.

[60] Qun Gao, P. forster,K. R. Mobus, G. S. Moschytz "Fingerprint recognition using CNNs: fingerprint preprocessing", IEEE International Symposium on Circuits and Systems, vol 3, 6-9 May 2001 pp. 433 - 436

[61] S. J. McKenna, S. Jabri, Z. Duric, H. Wechsler, and A. Rosenfeld, "Tracking groups of people," Computer Vision and Image Understanding: CVIU, vol. 80, no. 1, pp. 42–56, 2000.

[62] S. Lim and A. El Gamal, "Integrating image capture and processing beyond single chip digital camera," in Proc. SPIE Electronic Imaging 2001 Conf., San Jose, CA, Jan. 2001, vol. 4306.

[63] T. Komuro, I. Ishii, M. Ishikawa, and A. Yoshida, “A digital vision chip specialized for high-speed target tracking,” IEEE Trans. Electron Devices, vol. 50, no. 1, pp. 191–199, Jan. 2003.

[64] D. Handoko, K. S, Y. Takokoro, M. Kumahara, and A. Matsuzawa, “A CMOS image sensor for local-plane motion vector estimation,” in Symp. VLSI Circuits Dig. Papers, Jun. 2000, vol. 3650, pp. 28–29.

[65] R. Perfetti, E. Ricci, D. Casali, G. Costantini, ” Cellular Neural Networks With Virtual Template Expansion for Retinal Vessel Segmentation”, IEEE Transactions on Circuits and Systems II: Express Briefs, Feb. 2007, vol 54, No 2, pp. 141 - 145

[66] C.-Y.Wu and C.-T. Chiang, “A low-photocurrent CMOS retinal focalplane sensor with a pseudo-bjt smoothing network and an adaptative current schmitt trigger for scanner applications,” IEEE Sensors J., vol. 4, no. 4, pp. 510–518, Aug. 2004.

[67] Chao-Hui Huang, Chin-Teng Lin, "Bio-Inspired Computer Fovea Model Based on Hexagonal-Type Cellular Neural Network", IEEE Transactions On Circuits And Systems I: Regular Papers, January 2007, vol. 54, No. 1

[68] Chung Yu Wu, Wen Cheng Yen, .An efficient and compact integration of CMOS image sensors and cellular neural network (CNN) for intelligent processing”, IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems, 15-18 Ago. 1999 pp. 232 - 236

[69] D. Stoppa, A. Somoni, L. Gonzo, M. Gottardi, and G.-F. Dalla Betta, “Novel CMOS image sensor with a 132-db dynamic range,” IEEE J. Solid-State Circuits, vol. 37, no. 12, pp. 1846–1852, Dec. 2002.

[70] D. X. Yang, A. E. Gamal, B. Fowler, and H. Tian, “A 640×512 CMOS image sensor with ultrawide dynamic range floating-point pixellevel ADC,” IEEE J. Solid-State Circuits, vol. 34, no. 12, pp. 1821–1834, Dec. 1999.

[71] G. Andreou, K. A. Boahen, A. Pavasovic, P. O. Pouliquen, R. E. Jenkins, K. Strohbehn, “Current-mode subthreshold MOS circuits for analog VLSI neural systems”, IEEE Trans. on Neural Networks, Mar 1991, vol 2, pp. 205-213

[72] G. Chapinal, S. Bota, M. Moreno, J. Palacin, and A. Herms, “A "128 × 128 CMOS image sensor with analog memory for synchronous image capture,” IEEE Sensors J., vol. 2, no. 2, pp. 120–127, Apr. 2002.

[73] J. C. Gealow and C. G. Sodini, “A pixel parallel image processor using logic pitch-matched to dynamic memory,” IEEE J. Solid-State Circuits, vol. 34, no. 6, pp. 831–839, Jun. 1999.

[74] M. Sakakibara, S. Kawahito, D. Handoko, N. Nakamura, M. Higashi, K. Mabuchi, and H. Sumi, “A high-sensitivity CMOS image sensor with gain-adaptative column amplifiers,” IEEE J. Solid-State Circuits, vol. 40, no. 5, pp. 1147–1156, May 2005.

[75] M. Schwarz,R. Hauschild,B. J. Hosticka,J. Huppertz, T. Kneip, S. Kolnsberg, L. Ewe, Hoc Khiem Trieu, "Single-Chip CMOS Image Sensors for a Retina Implant System”, IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, Julio 1999, vol 46, No 7, pp. 870 - 877

[76] M. Stelzle, A. Stett, B. Brunner, M. Graf, W. Nisch, "Micro-Photodiode Arrays as Artificial Retina Implant: Electrical Properties", Proceedings of the 22 Annual EMBS International Conference, July 23-28,2000, Chicago IL

[77] R. Carmona-Galán, F. Jiménez-Garrido, C.M. Domínguez-Mata,R. Domínguez-Castro, S. Espejo Meana, I. Petras, A. Rodríguez-Vázquez, “Second-Order Neural Core for Bioinspired Focal-Plane Dynamic Image Processing in CMOS", IEEE Transactions On Circuits And Systems I: Regular Papers, May 2004, vol. 51, No 5

[78] R. M. Philipp et al., “A 128×128 33 mw 30 frames/s single-chip stereo imager,” in IEEE ISSCC Dig. Tech. Papers, 2006, pp. 506–507.

[79] S. Arakawa, Y. Yamaguchi, S. Akui, Y. Fukuda, H. Sumi, H. Hayashi, M. Igarashi, K. Ito, H. Nagano, M. Imai, and N. Asari, "A 512GOPS fullyprogrammable digital image processor with full HD 1080p processing capabilities," in Proc. IEEE ISSCC Dig. Tech. Papers, San Francisco, CA, Feb. 2008, p. 312.

[80] S. G. Smith, J. E. D. Hurwitz, M. J. Torrie, D. J. Baxtr, A. A. Murray, P. Likoudis, A. J. Holmes, M. J. Panaghiston, R. K. Henderson, S. Anderson, P. B. Denyer, and D. Renshaw, "A single-chip CMOS 306×244-pixel NTSC video camera and a descendant coprocessor device," IEEE J. Solid-State Circuits, vol. 33, no. 12, pp. 2104–2111, Dec. 1998.

[81] S. H. Hong and W. Yang, "An embeddable low power SIMD processor bank," in Proc. IEEE ISSCC Dig. Tech. Papers, San Francisco, CA, Feb. 2000, pp. 192–193.

[82] V. M. Brea, D. L. Vilarino, A. Paasio, D. Cabello, "Design of the processing core of a mixed-signal CMOS DTCNN chip for pixel-level snakes", IEEE Transactions on Circuits and Systems I: Regular Papers, May 2004, vol 51, No 5, pp. 997 - 1013

[83] Y. Chi, U. Mallik, E. Choi, M. Clapp, G. Gauwenberghs, and R. Etienne-Cummings, "Cmos pixel-level ADC with change detection," in Proc. Int. Symp. Circuits and Systems (ISCAS), May 2006, pp. 1647–1650.

[84] M. K. Hu, "Visual Pattern Recognition by Moment Invariants", IRE Trans. Info. Theory, vol. IT-8, pp.179–187, 1962

[85] J. Flusser: .°n the Independence of Rotation Moment Invariants", Pattern Recognition, vol. 33, pp. 1405–1410, 2000.

[86] J. Flusser and T. Suk, Rotation Moment Invariants for Recognition of Symmetric Objects", IEEE Trans. Image Proc., vol. 15, pp. 3784–3790, 2006.

[87] B. K. Khailany, T. Williams, J. Lin, E. P. Long, M. Rygh, D. W. Tovey, andW. J. Dally, "A programmable 512 GOPS stream processor for signal, image, and

video processing," IEEE J. Solid-State Circuits, vol. 43, no. 1, pp. 202–213, Jan. 2008.

[88] L. Lindgren, J.Melander, R. Johansson, and B.Moller, "A multiresolution 100-GOPS 4-Gpixels/s programmable smart vision sensor for multisense imaging," IEEE J. Solid-State Circuits, vol. 40, no. 6, pp. 1350–1359, Jun. 2005.

[89] M. Loinaz, K. Singh, A. Blanksby, D. Inglis, K. Azadet, and B. Ackland, "A 200 mv 3.3 V CMOS color camera IC producing "352×288" 24-b video at 30 frames/s," IEEE J. Solid-State Circuits, vol. 33, no. 12, pp. 2092–2103, Dec. 1998.