PRACTICA PROFESIONAL SUPERVISADA

*Procesamiento digital de imágenes por hardware: implementación en hardware de la operación de convolución en 2-D con reutilización dinámica de memoria*

### Alumnos: Casabella Martin, 39694763

Sulca Sergio, xxxxxxxx

Vignolles Ivan Maximiliano, xxxxxxxx

### Carrera: Ingeniería en Computación

### Tutor: Dr.Ing. Ariel L.Pola

**2018**

**Department of Research and Development Fundacion Fulgor**

Entonces, matemáticamente:

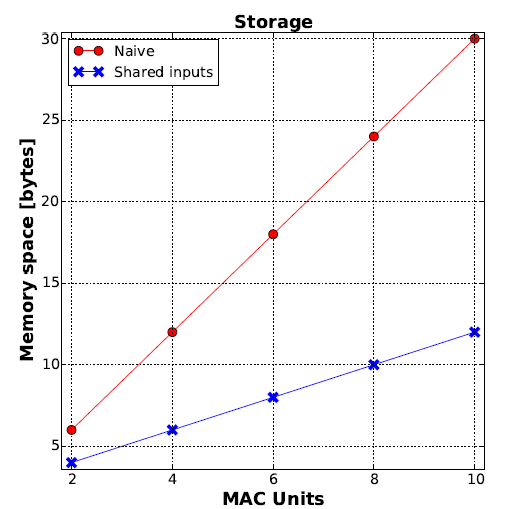
* Cuando es un múltiplo de , debe existir un numero entero , tal que:

Sintetizando el algoritmo descrito, dadas y un kernel de :

* Lote de imagen sin procesar, con columnas es necesario
* El lote procesado tendrá un ancho de columnas
* Las ultimas memorias no se sobrescriben
* Estas columnas se reutilizan como las primeras columnas del siguiente lote
* el ancho del lote transmitido se reduce a , con la excepción de que el primer lote mantiene un ancho de .
* Sea el periodo el número de iteraciones necesario para obtener la relación ente las columnas de memoria originales con respecto a las entradas de las unidades .

Cuando es un múltiplo de , debe existir un numero entero , tal que:

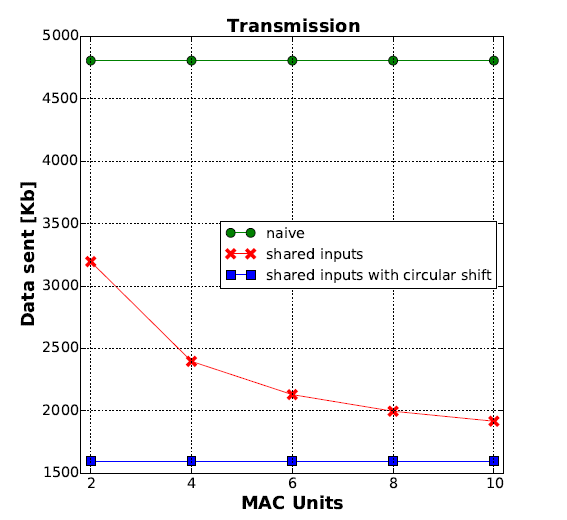
Veamos las ventajas del diseño planteado de forma gráfica:



*Cantidad de memoria requerida en función del grado de paralelismo*

Se ve en la figura que a medida que se instancian más unidades , o en otras palabras, a medida que se aumenta el grado de paralelismo del sistema haciendo referencia a la cantidad de convoluciones en paralelo con las que se trabaja, el primer enfoque propuesto requiere mucha mas memoria para operar que el enfoque de información compartida, donde se reutilizan ciertas columnas dada por la matemática y relaciones descritas.

Por otra parte, para lo que refiere a la transmisión, se tiene:



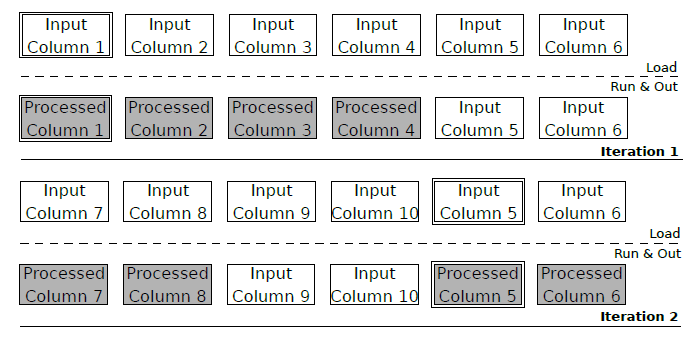
*Transmisión de datos en función de unidades MAC instanciadas*

Se mencionó que, debido al solapamiento, hay información repetida entre un lote recibido y el siguiente, por lo que, para reducir la transmisión de datos, esta información repetida se mantiene en memoria y solamente se transmite la parte faltante del lote entrante.

En la figura se muestra como el diseño planteado con el desplazamiento circular requiere una transmisión constante notablemente menor a comparación del primer enfoque que se planteo, y como también el hecho de reutilizar la información compartida tiende a una disminución en la transmisión de datos durante el funcionamiento del sistema. La imagen con la que se efectuaron las gráficas tenía un tamaño de .

Se analiza el caso de tener instanciadas , y un kernel con para clarificar la operación del algoritmo descrito.

El proceso de escritura en memoria, utilizando el desplazamiento circular de columnas mencionado, se puede apreciar en la siguiente figura:



Se ve en la figura, que hay una línea punteada que separa la etapa de carga y la de procesamiento y salida de datos.

Como consideramos un kernel con y s , :

* se necesitan o requieren columnas de memoria
* En la primera iteración, se cargan las memorias a con el nuevo lote (donde cada memoria instanciada alberga una columna del lote) y el resultado se almacena en las memorias a , marcadas con otro color y en la etapa *run & out.*
* En la segunda iteración, los datos del nuevo lote se cargan sin sobrescribir las memorias y . Luego del procesamiento, se almacenan los datos comenzando en la memoria hasta la última, para luego sobrescribir los datos del lote previo desde el principio.

La lógica del sistema se implementa en el bloque , que sirve como interfaz entre las memorias y el resto de los componentes, manteniéndolos independientes al grado de paralelismo del sistema y al número de iteración.

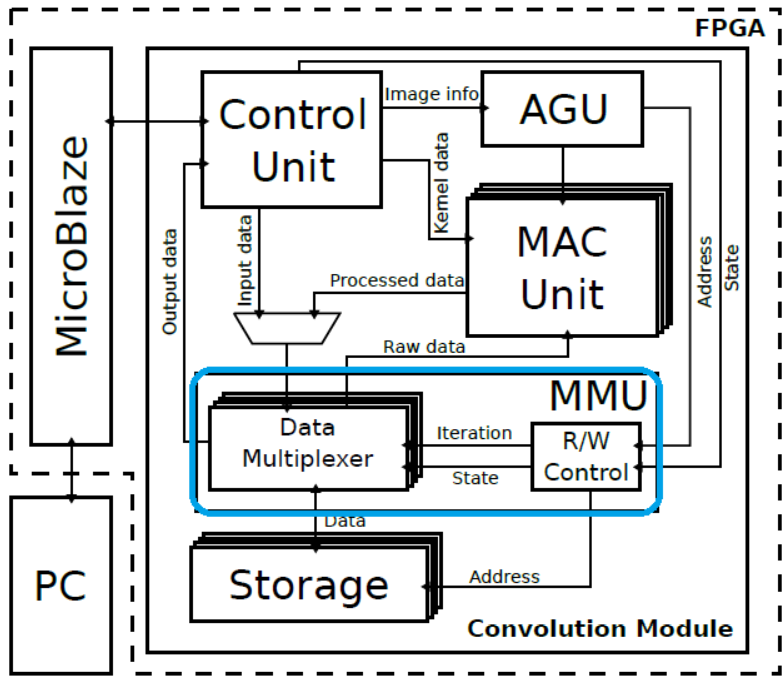
**)**

Este bloque, en cada iteración, mantiene un seguimiento de:

* Las posiciones de memoria donde el lote entrante debe ser almacenado
* la información que debe alimentar a cada unidad
* La posición de memoria donde la información procesada debe ser almacenada
* El orden en el cual la información procesada debe ser devuelta

Para cumplir con estas funciones, se basa en una máquina de estados finitos interna al módulo, donde cada estado corresponde a un conjunto de las posiciones citadas anteriormente. El número de estados está dado por el número de iteración de la ecuación:

Este bloque consta de un conjunto de multiplexores cuyas líneas o entradas de selección son manejadas por la máquina de estados mencionada, para así hacer el routing de la información entrante y saliente, como se ve en la figura resaltado en color azul:

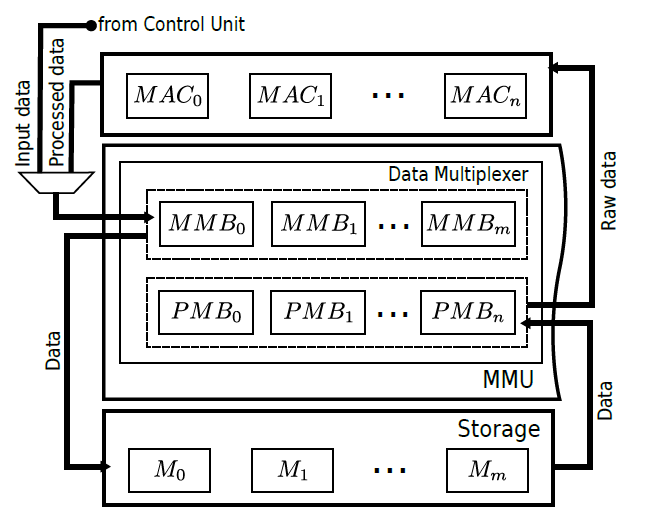


Analizando la arquitectura interna de los bloques que conforman este módulo, el bloque *data multiplexer* está conformado por un conjunto de bloques más pequeños clasificados en dos clases acorde a la función que cumplen:

Los hacen el routing de la información sin procesar (*raw data)* y la información desde las unidades hacia las memorias.

Los hacen el routing de la información desde las memorias hacia las unidades .

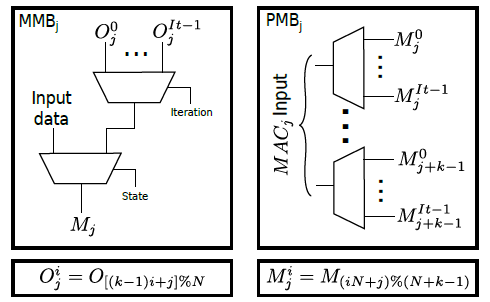
Este flujo se puede ver en la siguiente figura:



Tanto los bloques como los tienen un numero de entradas proporcional al numero de estados de la interna del bloque .

Las entradas a sus multiplexores se definen respectivamente:

El símbolo representa la operación modulo, y toma valores desde a .



# Características del proyecto

En el proyecto desarrollado se tienen las siguientes características:

* Implementación de la operación de convolución bidimensional en una FPGA
* Particionamiento de imagen por columnas, y carga de las mismas en la placa
* Procesamiento con cierto grado de paralelismo
* Kernel de 3x3, configurable
* Manejo de imágenes de escala de grises (cada píxel tiene asociado un valor de intensidad o escala de gris)
* Reutilización de hardware: específicamente, de Block RAM, uno de los elementos que mas energía consume en FPGAs.
* Funcionalidades y etapas en su totalidad: el sistema implementado cubre todas las etapas (carga, procesamiento, devolución )
* Escalabilidad
* Funcionalidades y etapas en su totalidad: el sistema implementado cubre todas las etapas (carga, procesamiento, devolución )
* Escalabilidad

Restricciones:

Debido al tamaño de la FPGA utilizada no se utilizo otra interfaz, como por ejemplo Ethernet o acceso directo a memoria (DMA). Pese a esta limitación, la arquitectura propuesta es indiferente a estos métodos.