# COMPUTADOR CON SISTEMA MULTIPROCESADOR HETEROGÉNEO

Arturo Salas Delgado
e-mail: artsaldel@gmail.com
Daniel Chacón Rojas
e-mail: dannychr27@gmail.com
Leonardo Araya Martínez
e-mail: leoam@hotmail.es

**ABSTRACT:** The objective of this project is to apply the concepts of multiprocessor systems and memory schemes, in designing a computer with heterogeneous multiprocessor that includes the interaction between a hard processor and a soft processor. This, in order to run an image filtering application, called maximum filter. The application will run in parallel between the two processors.

## 1. INTRODUCCIÓN

En el presente proyecto se deberán aplicar los conceptos de arquitectura de computadores, vista como una combinación de elementos de software y hardware, en el diseño e implementación de un computador basado en un sistema multiprocesador, con una interfaz de memoria compartida y sincronización entre los procesadores. Esto con el fin de desarrollar una aplicación de procesamiento digital de imágenes, en la que cantidad de datos a procesar crea la necesidad de un sistema multiprocesador, con el fin de lograr un mayor desempeño en la ejecución. Durante el desarrollo del proyecto se diseñaron una serie de interfaces de memoria compartida y sincronización, así como diseño, integración y programación de sistemas computacionales en general. Adicionalmente, se incluyó en la FPGA los módulos de memoria, controlador de PCI y la lógica de sincronización para uso de memoria compartida. En la figura 1 se muestra un diagrama de bloques de la organización del sistema.

## 2. SISTEMA DESARROLLADO

### 2.1 Hardware

| Name | Description |
|---|---|
| ⊞ **clk_50** | Clock Source |
| ⊞ **pcie_ip** | IP_Compiler for PCI Express |
| ⊞ **sgdma** | Scatter-Gather DMA Controller |
| ⊞ **nios2** | Nios II Processor |
| ⊞ **onchip_memory** | On-Chip Memory (RAM or ROM) |

Figura 2. Componentes del hardware del sistema desarrollado.

La parte de hardware del proyecto se compone de los módulos presentes en la figura anterior [1]. Estos son:

- Clock de 50Mhz: Este se encarga de darle sincronía a todo el sistema por medio de una señal de clock.
- PCI: Es el bus encargado de conectar los diferentes periféricos del sistema.
- DMA: Este módulo se utiliza para enviar datos masivos a la memoria RAM.
- NIOS: Es el procesador secundario en el cual se realiza, paralelamente, el procesamiento.
- Onchip memory: Es la memoria RAM con la que cuenta el sistema.

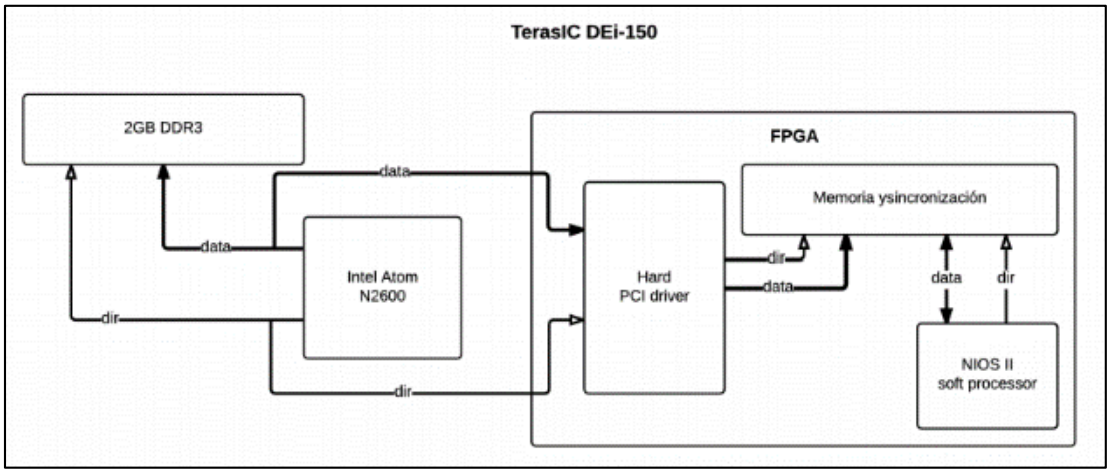


Figura 1. Diagrama de bloques de la organización del sistema desarrollado.

### 2.2 Software

La parte de software del proyecto se puede dividir en dos grandes partes: un módulo de comunicación y un filtro de dilatación.

El módulo de comunicación se encarga de dos cosas: la primera es habilitar la conexión entre el procesador Atom y el procesador Nios, de forma que se pueda realizar el procesamiento paralelo entre los dos procesadores. La otra función que tiene esta clase es enviar los datos, desde el procesador Atom, que deberá computar el procesador Nios.

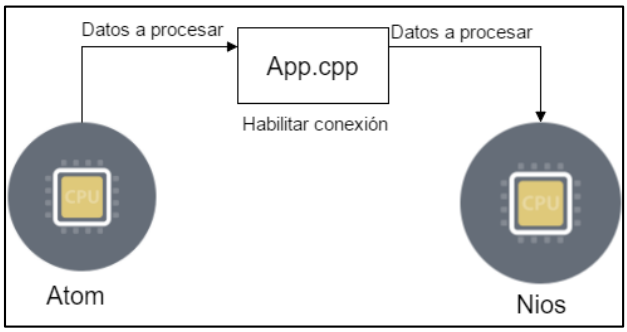


Figura 3. Diagrama de arquitectura de alto nivel de la clase app.cpp.

El otro componente del software es un filtro morfológico de máximo o filtro de dilatación, que corresponde a una operación básica en el procesamiento digital de imágenes. Para la transformación de la imagen a escala de grises se utiliza OpenCV, ya que "ha sido diseñado para ser eficiente en cuanto a gasto de recursos computacionales y con un enfoque hacia las aplicaciones de tiempo real"[2]. En este tipo de filtro, se posee un elemento estructural simétrico, conocido como kernel. Dicho kernel se compone de una cantidad de 5x5 pixeles. Para el filtro de máximos, el kernel recorre pixel por pixel la imagen y asigna al pixel actual (pixel central del kernel) el valor máximo de todos los pixeles de la imagen que se encuentren dentro del kernel. Un ejemplo de la aplicación del filtro se puede observar en la figura 4.

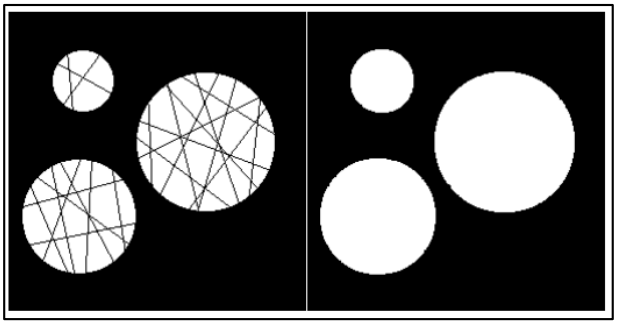

Figura 4. Ejemplo de aplicación del filtro de máximo.

Es importante recalcar que hay una sincronización para que los dos procesadores puedan trabajar paralelamente, para esto se usan hilos de ejecución, que "son una secuencia de instrucciones que pueden ser ejecutadas concurrentemente con otras secuencias en entornos multiproceso, mientras que comparten un mismo espacio de direcciones" [3].

## 3. RESULTADOS

A continuación se presentan dos ejemplos prácticos de los resultados obtenidos: una imagen de 960x720 pixeles y otra de 220x225 pixeles antes y después de aplicar el algoritmo. Nótese que, tal como lo requiere el algoritmo, la imagen de salida será siempre en escala de grises.

**Primer ejemplo**

La figura 5 muestra 3 imágenes: la imagen de la izquierda es la imagen original que el usuario ingresa en la aplicación realizada. Esta aplicación genera dos imágenes, una es la imagen transformada a escala de grises (imagen del centro) y la otra es la imagen a escala de grises aplicando el algoritmo de máximo (imagen de la derecha).

Figura 5. Resultados de la aplicación del filtro de máximo ejemplo 1.

Se realiza una prueba de medición de tiempos de la aplicación aplicando diferentes cantidades de iteraciones, igualmente, usando la imagen de la figura anterior. Se denota que conforme aumentan las iteraciones, el tiempo va a ser cada vez mayor. Además, conforme se distribuya el peso de ejecución a cada procesador, los tiempos son diferentes, tal como se muestra en la figura 6:

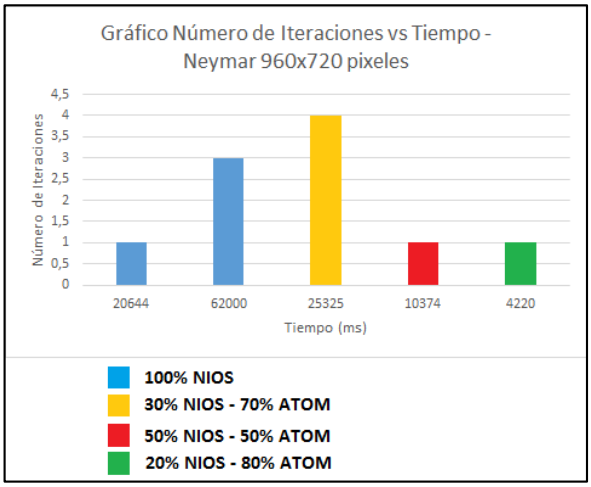


Figura 6. Gráfica de distribución de tiempo del ejemplo 1.

De la figura anterior, también se denota que el peso o porcentaje de trabajo que se le asigna a cada procesador influye directamente en el tiempo de ejecución, o sea, cuanto más procesamiento se le asigna al procesador esclavo (NIOS), más tiempo va a durar en procesar la imagen.

**Segundo ejemplo**

En este ejemplo se realiza el mismo proceso que el ejemplo anterior. A continuación (figura 6)se muestra la imagen utilizada para el ejemplo, la imagen transformada a escala de grises y la imagen a la cual se le aplica el algoritmo de máximo:

Y el siguiente gráfico demuestra la distribución de tiempos que se dan al momento de ejecutar la imagen para una cantidad de iteraciones diferente y para una distribución de trabajo diferentes:

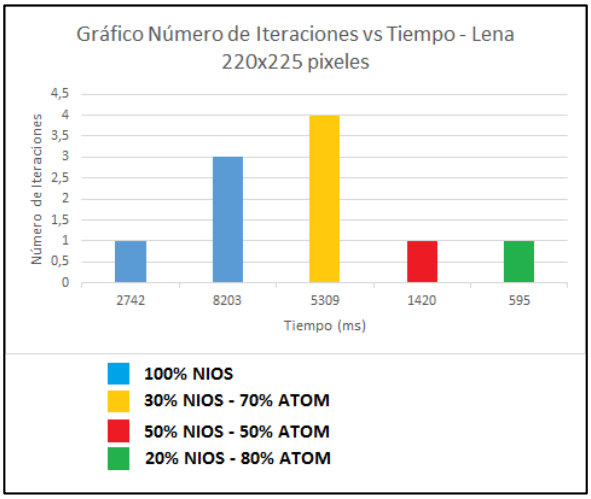


Figura 7. Gráfica de distribución de tiempo del ejemplo 2.

## 4. CONCLUSIONES

Basado en los resultados experimentales se puede concluir que el tiempo de ejecución va a ser proporcional al porcentaje de procesamiento que se le asigne al procesador esclavo. Esto se debe al hecho de que el procesador principal es más rápido que el procesador esclavo. Esto porque el enviar datos al procesador esclavo (Nios) requiere su tiempo y esto afecto en el rendimiento del sistema. Si ambos procesadores fueran iguales el tiempo de ejecución sería inversamente proporcional al porcentaje de procesamiento del procesador esclavo.

Figura 6. Resultados de la aplicación del filtro de máximo ejemplo 2.

Cuando se ejecuta un mismo programa una cantidad n de veces, cada vez el tiempo de ejecución será menor. Esto se debe al hecho de que la memoria caché se llena con los datos necesarios para ejecutar el programa. Si se desea hacer una medición de tiempos real se debe tomar en cuenta este factor y, después de cada iteración, limpiar la memoria caché.

## 5. REFERENCIAS

[1] Altera, «Altera,» [En línea]. Available: https://www.altera.com/products/design-software/fpga-design/quartus-prime/features/qts-qsys.html. [Último acceso: 1 6 2016].

[2] A. Rodriguez, «Universidad de La Laguna,» 18 5 2015. [En línea]. Available: https://osl.ull.es/software-libre/opencv-libreria-vision-computador/. [Último acceso: 2 6 2016].

[3] cplusplus, «cplusplus,» [En línea]. Available: http://www.cplusplus.com/reference/thread/thread/. [Último acceso: 29 5 2016].