Diseño e Implementación de un ASIP vectorial para filtros clásicos de imágenes

Esteban Alvarado Vargas*, Olman Castro Hernández†, Grabiel Abarca Aguilar‡
Área Académica de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica
Email: * estalvgs1999@estudiantec.cr, † olcastro@estudiantec.cr, ‡ ,

***Abstract*—Classic Filters is something used on Image Processing and classic image filters relate to complex algorithms nowadays. In this article we explain how with a custom Vectorial Processor you can apply Filters to 256 x 256 images. Sepia, Negative and Gray Scale filters are used for this example and the advantages of using this processor for this algorithm are shown in the process.**

***Palabras clave*—Vectorial Processor, 8 Lanes, Sepia, Gray Scale, Negative, Vectorial ALU.**

I. Introducción

El paralelismo en múltiples niveles ha sido la motivación del diseño de computadoras con el fin de mejorar el rendimiento términos de energía y costo. Un tipo básico de paralelismo es a nivel de datos donde surge porque hay muchos datos que pueden ser operados al mismo tiempo. Las arquitecturas vectoriales son los referentes en este aspecto porque utilizan el concepto de SIMD para explicar el paralelismo a nivel de datos aplicando una única instrucción a una colección de datos de manera paralela.

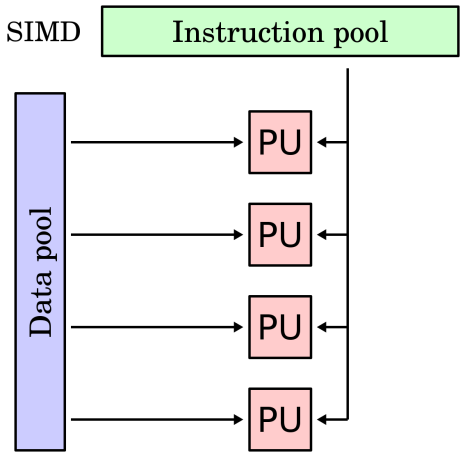

Fig. 1: Diagrama SIMD [1]

El proyecto aplicará los principios arquitectura de computadoras para el diseño e implementación de un procesador vectorial. La arquitectura de un set de instrucciones (ISA) con el fin de aplicarla para diversos algoritmos de filtros de imágenes.

En el proyecto se desarrollará un acercamiento práctico al diseño de un set de instrucciones propio y específico, la realización en hardware de un modelo funcional de un procesador vectorial que implemente el set propuesto, y programación de sistemas computacionales en general mediante un compilador encargado de interpretar los resultados del procesador y finalmente un sistema encargado de mostrar los resultados obtenidos.

En el presente documento, se expondrá al lector los conceptos principales para el entendimiento de este mediante el marco teórico, como fue desarrollado el proyecto en la sección procesador desarrollado, los resultados obtenidos y finalmente las conclusiones de este.

II. Marco Teórico

A continuación se explican la teória investigada para desarrollar el proyecto:

A. Filtro Sepia

Para realizar el filtro sepia se necesita crear un algoritmo donde intercambie el valor de los pixeles de acuerdo a una tonalidad marrón. Para esto se realiza mediante la ecuación de la figura 2.

$$r_{out} = 0.393r_{in} + 0.769g_{in} + 0.189b_{in}$$
$$g_{out} = 0.349r_{in} + 0.686g_{in} + 0.168b_{in}$$
$$b_{out} = 0.272r_{in} + 0.534g_{in} + 0.131b_{in}$$

Fig. 2: Ecuación Sepia[2]

B. Filtro Escala de Grises

Como se indica en Gimp [3], la aplicación de la tonalidad gris a una imagen se puede realizar por medio de multiples formas, por brillo, luminosidad y promedio, sin embargo la ecuación más fácil de aplicar para este algoritmo es por medio de promedio. La cual indica que $r_{out}, g_{out}, b_{out} = avg(r_{in}, g_{in}, b_{in})$.[3]

C. Filtro Negativo

Como se indica en Geeky Hacker [4], para la aplicación del Filtro Negativo se realiza por medio de una resta del valor máximo con el color. En este caso el Valor máximo sería 255,

ya que todos los colores llegan como máximo a este valor. En resumen $pixel = 255 - pixel_{in}$).[4]. Esto invierte la paleta de colores, cambiando el color, por el color complementario.

III. Procesador desarrollado

La presente sección, expone la conformación del procesador desarrollado. En la Fig. 3 se puede observar el diagrama de primer nivel, este se encuentra dividido por tres bloques. El CPU que almacena la lógica principal del procesador y la memoria de instrucciones y datos bloques externos de este.

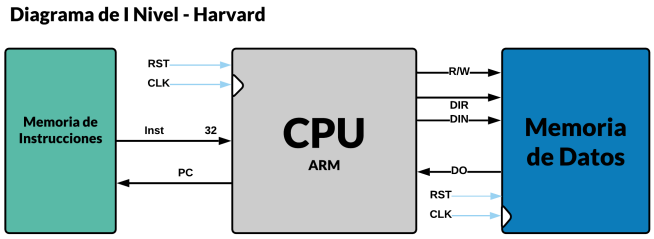

Fig. 3: Diagrama de primer nivel

En la Fig. 4, podemos observar como el bloque del CPU se descompone ahora entres más modulares, en donde, podemos encontrar la unidad de control y unidad de detección de riesgos, módulos encargados de la detección de conflictos en la microarquitectura del sistema.

En caso del módulo de ruta de datos profundizaremos más en el siguiente nivel.

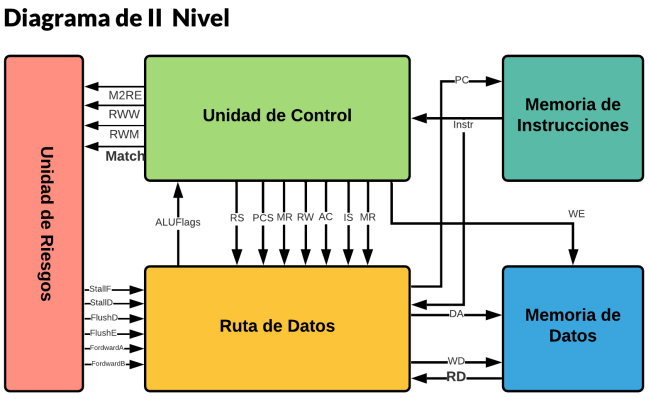

Fig. 4: Diagrama diagrama de segundo nivel

En el caso del diagrama de tercer nivel Fig. 5, podemos ver como la ruta de datos sufre una fragmentación si la comparamos con una arquitectura no vectorial. Esto sucede ya que las instrucciones y operaciones vectoriales necesitan unidades de procesamiento especial. Debido a eso, la lógica luego del fetch debe de dividirse. Cada sección posee su propio decodificador, AlUs y memoria (al final todo debe almacenarse en la memoria vectorial).

Finalmente, si profundizamos un poco más en el diagrama de tercer nivel, tal como muestra la Fig. 6. Es posible observar

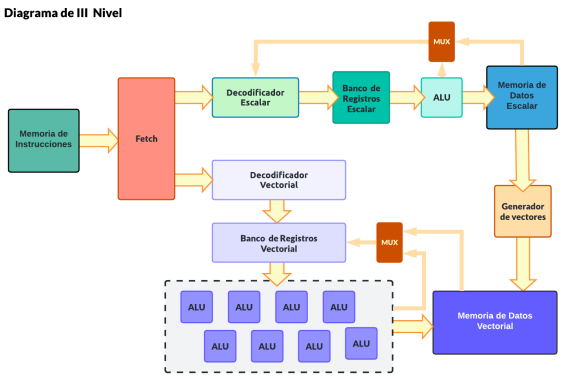

Fig. 5: Diagrama de tercer nivel

como en lugar de una unidad de procesamiento lógico como en la sección escalar poseemos ocho, esto con el fin de poder paralelizar las operaciones que son necesarias en el vector y estos resultados viajan finalmente por el bus de datos a la memoria vectorial.

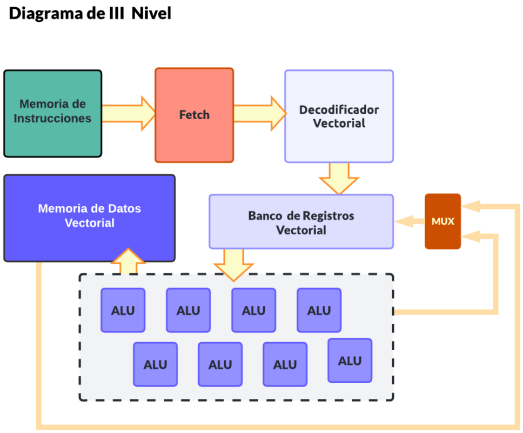

Fig. 6: Diagrama de tercer nivel sección vectorial

IV. Resultados

Como se mostró anteriormente, los algoritmos fueron aplicados a una imágenes de 256 x 256 pixeles, esta imágen se puede ver en la figura 7. Este procesamiento se realizó por medio de Alu Vectorial, que se puede mostrar su funcionamiento en el Testbench de la figura 8. El algoritmo realizaba tres filtros, se va a analizar cada uno con el resultado obtenido.

A. Filtro Negativo

El filtro negativo obtuvo como resultado la imagen de la figura 9. Esta muestra los colores invertidos con respecto a su complemento en la paleta de colores. Para este efecto se usó el algoritmo explicado en el Marco Teórico sin variaciones.

Fig. 7: Imagen de Ejemplo

Fig. 9: Imagen aplicada por Filtro Negativo

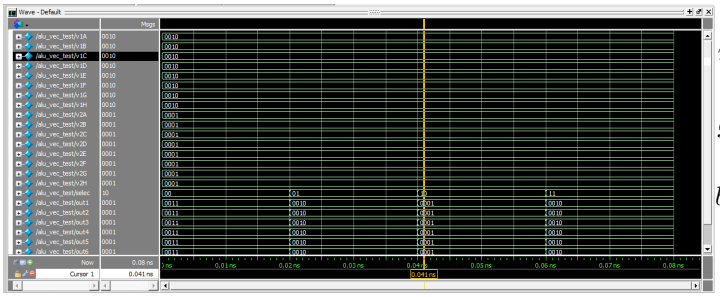

Fig. 8: Testbench de la ALU vectorial

$$r_{out} = 0.393r_{in} + 0.769g_{in} + 0.189b_{in} \approx \frac{r_{in}}{3} + g_{in} + \frac{b_{in}}{5}$$

$$g_{out} = 0.349r_{in} + 0.686g_{in} + 0.168b_{in} \approx \frac{r_{in}}{3} + g_{in} + \frac{b_{in}}{6}$$

$$b_{out} = 0.272r_{in} + 0.534g_{in} + 0.131b_{in} \approx \frac{r_{in}}{4} + \frac{g_{in}}{2} + \frac{b_{in}}{8}$$

Fig. 10: Ecuación Sepia

B. Filtro Sepia

En el caso del filtro sepia se obtuvo el resultado de la fígura 11. Esta muestra una tonalidad de los colores marrones o amarillentos. Se realizó una variación ya que el procesador no posee unidad de punto flotante, esta es la de la ecuación 10.

C. Filtro Escala de Grises

En el caso del filtro sepia se obtuvo el resultado de la fígura **??**. Esta muestra una tonalidad de los colores grises, solo tonos de blanco y negro. Se realizó una variación ya que el procesador es de 8 bits, esta es la de la ecuación 12.

V. Conclusiones

Como se demostró anteriormente la utilidad de un procesador vectorial es bastante grande y se puede conlcuir que es una herramienta necesaria para el desarrollo de computadores ya que permite la manipulación de datos a en cantidades grandes. También se puede concluir su utilidad en el campo gráfico ya que este puede implementar con facilidad programas que manejen imágenes. Cómo último, su utilidad es cómo la de un GPU, lo que puede llegar a concluir que son bastante útiles en el campo del procesamiento gráfico.

References

[1] Arstechnica (J. Stokes). “SIMD architectures”, 2000. [Online]. Available in: https://arstechnica.com/features/2000/03/simd/ [Accessed: 18 May 22].

[2] García, Y., 2022. Yabir’s blog. [online] Yabirgb.com. Available at: ¡https://www.yabirgb.com/sepia_filter/¿ [Accessed 9 April 2022].

[3] Docs.gimp.org. 2022. 5.10. Desaturate. [online] Available at: ¡https://docs.gimp.org/2.8/en/gimp-tool-desaturate.html¿ [Accessed 9 April 2022].

[4] Geeky Hacker. 2022. Image processing (Negative algorithm). [online] Available at: ¡https://www.geekyhacker.com/2013/05/03/image-processing-negative-algorithm/¿ [Accessed 9 April 2022].

Fig. 11: Imagen aplicada por Filtro Sepia

$$r_{out} = g_{out} = b_{out} = \frac{r_{in} + g_{in} + b_{in}}{3} = \frac{r_{in}}{3} + \frac{g_{in}}{3} + \frac{b_{in}}{3}$$

Fig. 12: Ecuación Escala de Grises

Fig. 13: Imagen aplicada por Filtro Escala de Grises