TEC | Tecnológico de Costa Rica

Instituto Tecnológico de Costa Rica
Área Académica de Ingeniería en Computadores
CE 4302 — Arquitectura de Computadores II

Proyecto Grupal 1:
Diseño e Implementación de un ASIP vectorial para composición alfa

Fecha de asignación:	28 de abril	Fecha de entrega:	26 de mayo
# Int. (# Grupos):	3 (2) y 4 (5)	Profesores:	Luis Barboza Artavia

Mediante el desarrollo de este proyecto, el estudiante aplicará los conceptos de paralelismo a nivel de datos, específicamente procesadores *Single Instruction Multiple Data* (SIMD) del tipo vectorial. Se realizará un *Application Specific Instruction Set Processor* (ASIP) por medio de una composición del canal alfa.

1. Atributos relacionados

A continuación se describen los atributos del graduado que se pretenden abordar con el desarrollo del proyecto.

1.1. Diseño (DI)

Capacidad para diseñar soluciones de problemas complejos de ingeniería, con final abierto y diseñar sistemas, componentes o procesos que cumplan con necesidades específicas, considerando la salud pública, seguridad, estándares pertinentes, así como los aspectos culturales, sociales, económicos y ambientales.

El atributo de diseño será evaluado tanto formativamente (reuniones de seguimiento con el profesor) como sumativamente, en especial en la sección de documentación de diseño de los entregables.

2. Descripción General

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. Los diseñadores han encontrado que SIMD trae ventajas puesto que es potencialmente eficiente en energía que otras técnicas como MIMD.

Para este proyecto se aplicarán los conceptos de arquitectura de computadoras para el diseño e implementación de un procesador vectorial. La arquitectura del set de instrucciones (ISA) será propuesto por cada grupo según las necesidades de la aplicación para realizar una composición del canal alfa. Los pasos que deben realizarse son los siguientes:

- Generar un degradado a color con orientación horizontal, vertical, diagonal o uno propuesto por el grupo[1]. El usuario elegirá los colores de entrada para formar el degradado.
- Con el degradado creado, el usuario podrá elegir la transparencia del mismo para ser aplicado a una imagen de entrada.

La implementación de este algoritmo se observa en la Figura 1.

Figura 1: Implementación del algoritmo para composición alfa.

2.1. Generación del degradado

Un degradado es una transición de colores que ocurre de manera lineal determinado por un ángulo o una dirección. Esto va a generar que la intensidad de cada canal sea dependiente de la dirección. Por ejemplo, para realizar un degradado horizontal del color negro al rojo, los canales G y B deben estar en 0, mientras que el R se determinará por la posición x. La figura 2 muestra el resultado del degrado.

Figura 2: Degradado de negro a rojo con orientación horizontal.

Para realizar un degrado vertical, la variable en los canales será la y. Por ejemplo, para hacer un degradado de azul a negro con orientación vertical, los canales R y G estarán en 0. Por su parte, el canal B se determinará por la posición y. El resultado de este ejemplo se muestra en la figura

[1]Pueden utilizar esta página web para probar los degradados

Figura 3: Degradado de azul a negro con orientación vertical.

Por último, un degradado diagonal es la combinación de los dos anteriores. Un canal debe ser determinado por la posición y, mientras que otro por la posición x. Por ejemplo, la figura 4 muestra un degradado donde el canal R se mantiene en 0, el G se determina por la posición x y el canal B por la posición y.

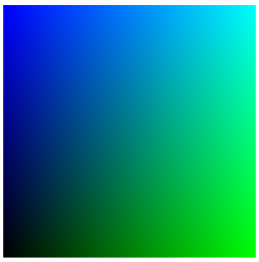

Figura 4: Degradado diagonal.

2.2. Composición alfa

La composición alfa es una técnica para combinar dos o más imágenes mediante el uso de un cuarto canal. La idea es utilizar la transparencia de una de las dos imágenes para mezclarlas en una sola imagen. En una representación de 8 bits, un 0 indica que la imagen es completamente transparente, mientras que en 255 la imagen es completamente opaca. Para este proyecto, la imagen de referencia será opaca, mientras que el degradado presentará un grado de transparencia definido por el usuario.

Para realizar la composición necesitaremos la transparencia del degradado (α_{in}) y los canales RGB de ambas imágenes. A continuación se detallan los cálculos para determinar el nuevo valor de los canales RGB.

$$R_{out} = R_{in1} \times (1 - \alpha_{in}) + R_{in2} \times \alpha_{in} \tag{1}$$

$$G_{out} = G_{in1} \times (1 - \alpha_{in}) + G_{in2} \times \alpha_{in} \tag{2}$$

$$B_{out} = B_{in1} \times (1 - \alpha_{in}) + B_{in2} \times \alpha_{in} \tag{3}$$

3. Especificación

Se le solicita desarrollar una **arquitectura** y una **microarquitectura** que realice el proceso de generación de un degradado y composición alfa. Se usará una imagen de entrada libre con dimensión mínima de 200×200. En la salida se mostrará la imagen final luego de realizada la composición.

Se deben seguir los siguientes requisitos generales de funcionalidad:

1. El diseño completo debe poder ser sintetizable en una tarjeta de desarrollo Terasic DE1-SoC-M TL2 (debe caber todo ahí, inclusive la imagen de entrada y el producto de la composición).

2. Las imágenes de entrada y salida deben ser almacenadas en memoria (se recomienda usar el bloque IP de la biblioteca de Quartus).

3. El sistema debe permitir la interacción con el usuario, para poder escoger la intensidad del rojo (0 %, 25 %, 75 % y 100 %), verde (0 %, 25 %, 75 % y 100 %), azul (0 %, 25 %, 75 % y 100 %), transparencia degradado (0 %, 25 %, 75 % y 100 %) y orientación (horizontal, vertical, diagonal y propuesto) mediante algún periférico (e.g., botones, switches).

4. La imagen mostrada debe ser escrita en un `.img` que será leído e interpretado por un software de alto nivel libre.

5. El ISA debe ser eficiente y congruente, con criterios de diseño definidos. Es importante hacer reuniones con el profesor para guía.

6. El formato de las imágenes será en escala de colores con píxeles con valores entre [0, 255].

Requisitos de Arquitectura ISA:

1. Debe diseñar un conjunto de instrucciones y arquitectura que permita solucionar el problema planteado, considerando detalles como:

   a) Modos de direccionamiento.

   b) Tamaño y tipo de datos.

   c) Tipo y sintaxis de las instrucciones.

   d) Registros disponibles y sus nombres.

   e) Codificación y descripción funcional de las instrucciones

Tome en cuenta que estos detalles deben ser justificados desde el punto de vista de diseño (complejidad, costo, área, recursos disponibles).

2. Las instrucciones a desarrollar son libres así como el tipo de datos. Aunque no hay un límite en cuanto la cantidad de instrucciones, es importante que provea al menos instrucciones para control de flujo, operaciones aritméticas-lógicas, acceso a memoria.

3. Los productos finales de esta etapa son el *instruction reference sheet* o *green sheet*.

4. El ISA debe ser personalizado y realizado por los estudiantes, no se aceptarán ISAs ya diseñados (e.g., ARM, x86, RISC-V, otros). Debe justificar cada característica del mismo.

Requisitos de Microarquitectura

1. La implementación diseñada debe ser correcta respecto a las reglas definidas por la arquitectura, esto quiere decir que el procesador debe ser capaz de ejecutar todas las instrucciones definidas y su especificación respecto a errores y excepciones.

2. El procesador diseñado debe emplear pipelining. Tenga en cuenta las implicaciones respecto a riesgos de dicha técnica, el uso de registros y unidades de ejecución. **No se revisará si no tiene pipeline.**

3. Debe ser implementado usando SystemVerilog.

4. No se permite realizar módulos especializados de hardware. Es un curso de Arquitectura de Computadores, no de Diseño de Sistemas Digitales.

5. El procesador debe tener capacidad de segmentación de memoria en datos e instrucciones además debe ser capaz de acceder los dispositivos de entrada y salida del sistema (GPIO, volcado de memoria, switches, etc).

6. Cada unidad funcional del sistema debe ser debidamente probada en simulación, para verificar su funcionamiento correcto (unit tests). Además debe incluir pruebas de integración y sistema. Se le solicita un plan de pruebas donde especifique los objetivos y descripción de las pruebas junto con sus resultados.

7. El procesador debe poseer al menos 4 *lanes* para ejecutar de manera paralela las operaciones en los vectores.

8. Los resultados finales de esta etapa son:

   a) El código fuente (SystemVerilog) y el bitstream para programar la tarjeta de desarrollo.

b) Un diagrama de bloques de la microarquitectura y descripción de las interacciones entre ellos.

c) Simulaciones de las pruebas unitarias y de integración.

d) Reporte de consumo de recursos del FPGA para el modelo.

Requisitos de Software:

1. Crear una aplicación (software) empleando la arquitectura diseñada, con el fin de implementar la aplicación descrita.

2. Debe realizar un programa ('compilador') que permita traducir las instrucciones del ISA a binario, con la finalidad de ejecutarlo en el procesador. No es necesario que realice análisis léxico, sintáctico y semántico (este curso no es de Compiladores).

El proceso de diseño debe incluir propuestas y comparación de viabilidad de las mismas.

4. Evaluación y entregables

La defensa será el mismo día de la entrega y todos los archivos (incluyendo código fuente) serán entregados a las 11:59 pm ese mismo día (realícenlo progresivamente y no lo deje para el final). La evaluación del proyecto se da bajos los siguientes rubros contra rúbrica correspondiente:

- Presentación proyecto 100 % funcional (65 %): La defensa se realizará de la siguiente manera: Debido a la situación actual (COVID-19) no se puede tener acceso a hardware u otros instrumentos y medios que requieran presencia y contacto físico tanto entre el profesor como los estudiantes. Por esta razón para la defensa se debe presentar lo siguiente en una hora:

  1. Todo el diseño debe ser sintetizable en una tarjeta: **Terasic DE1-SoC-M TL2**. Es decir, debe llegar hasta la generación de un **.sof**. Se garantiza que la tarjeta tenga suficientes recursos para almacenar y ejecutar el diseño según el reporte.
  2. Debe reservar los espacios de memoria para la imagen de salida.
  3. Mediante ModelSim debe crear un *testbench* de los mismos archivos de SystemVerilog que se usaron para sintetizar. Con este *testbench* debe escribir un archivo:
     - Imagen de salida (.img).

     Este archivo es una representación que contiene los valores de los pixeles.
  4. Debe crear un script de alto nivel para visualizar la imagen de entrada y la de salida.

  Los entregables adicionales que se revisarán en la defensa son los siguientes:

1. Arquitectura:
   a) *Instruction reference sheet* o *green sheet*.
2. Microarquitectura:
   a) Diagrama de bloques de la microarquitectura.
   b) Reporte de consumo de recursos del FPGA.
3. Software
   a) Programa de software.
   b) Compilador usado.
4. Plan de pruebas: debe incluir las simulaciones de las pruebas unitarias de los diferentes módulos.

- Artículo científico tipo *paper* (17.5 %): El paper a realizar deberá tener una extensión no mayor a 4 páginas completas (incluyendo bibliografías), deberá ser realizado con LaTeX, siguiendo un formato establecido (IEEE Transactions o ACM, por ejemplo). Se les provee un ejemplo de paper en el enlace. En general el *paper* deberá contar con las siguientes secciones:

  1. Abstract (en inglés): Un buen abstract tiene las siguientes características:
     a) Un abstract permite a los lectores obtener la esencia o esencia de su artículo o artículo rápidamente, para decidir si leer el artículo completo.
     b) Un abstract prepara a los lectores para seguir la información detallada, los análisis y los argumentos en su artículo completo.
     c) Un abstract ayuda a los lectores a recordar puntos clave de su paper.
     d) Un abstract es de entre 150 y 250 palabras.
  2. Palabras clave significativas (a lo sumo 6).
  3. Introducción: Una buena introducción muestra el contexto del problema o lo que se va a solucionar, introduce el tema al lector. Al final de la introducción se indica la organización del documento (primero se muestra el algoritmo, luego....).
  4. Algoritmo desarrollado.
  5. Resultados.
  6. Conclusiones escritas en prosa.
  7. Bibliografía, en formato IEEE y referenciadas en el texto (usar cite). Referencia bien para evitar problemas de plagio. Una documento no referenciado en el texto no existe.

- Documentación de diseño (17.5 %): Este documento se encuentra directamente ligado con el atributo DI. La documentación del diseño deberá contener las siguientes secciones:

1. Listado de requerimientos del sistema: Cada estudiante deberá determinar los requerimientos de ingeniería del problema planteado, considerando partes involucradas, estado del arte, estándares, normas, entre otros.
2. Elaboración de opciones de solución al problema: Para el problema planteado deberán documentarse al menos dos opciones de solución. Cada solución deberá ser acompañada de algún tipo de diagrama.
3. Comparación de opciones de solución: Se deberán comparar explícitamente las opciones de solución, de acuerdo con los requerimientos y otros aspectos aplicables de salud, seguridad, ambientales, económicos, culturales, sociales y de estándares.
4. Selección de la propuesta final: Se deberá evaluar de forma objetiva, válida y precisa las soluciones planteadas al problema y escoger una solución final.
5. Archivo tipo README donde especifiquen las herramientas que usaron. **Es un documento README.MD aparte.**

Se seguirán los siguientes lineamientos:

1. Los documentos serán sometidos a control de plagios para eliminar cualquier intento de plagio con trabajos de semestres anteriores, actual o copias textuales, tendrán nota de cero los datos detectados. Se prohíbe el uso de referencias hacia sitios no confiables.
2. No coloque código fuente en los documentos, quita espacio y aporta poco. Mejor explique el código, páselo a pseudocódigo o use un diagrama.