Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computadores

Programa de Licenciatura en Ingeniería en Computadores

Curso: CE-4302 Arquitectura de Computadores II

Especificación Proyecto 02: Diseño e Implementación de una Arquitectura de Dominio Específico (DSA) para *Downscaling* de Imágenes mediante Interpolación Bilineal Paralela

Profesores:

Luis Alonso Barboza Artavia

Ronald García Fernández

Fecha de entrega: 23-24 de octubre, 2025

Semestre: II 2025

Objetivo general

Diseñar, implementar y verificar una arquitectura de dominio específico (DSA) en FPGA que realice reducción de imágenes mediante interpolación bilineal, aplicando paralelismo a nivel de datos (DLP) en configuraciones secuencial y vectorial *SIMD*, con control e interfaz de comunicación para validación y análisis de rendimiento.

Atributos relacionados: Diseño (DI).

Definición:

Diseña soluciones creativas para problemas de ingeniería complejos y diseña sistemas, componentes o procesos para satisfacer las necesidades identificadas con la consideración adecuada para la salud y la seguridad públicas, el costo total de la vida, el carbono neto cero, así como las consideraciones de recursos, culturales, sociales y ambientales según sea necesario.

Indicadores

DI1- Identifica las necesidades y los requerimientos de un problema complejo de ingeniería considerando la salud y la seguridad pública, el costo total de la vida, el carbono neto cero, así como aspectos relacionados con recursos, culturales, sociales y ambientales según sea necesario.

DI2- Valora alternativas de solución para un problema complejo de ingeniería que cumplan con necesidades específicas, considerando la salud y la seguridad pública, el costo total de la vida, el carbono neto cero, así como aspectos relacionados con recursos, culturales, sociales y ambientales según sea necesario.

DI3- Diseña de forma creativa, la alternativa seleccionada que cumpla con las necesidades específicas para resolver el problema complejo de ingeniería, considerando la salud y la seguridad pública, el costo total de la vida, el carbono neto cero, así como aspectos relacionados con recursos, culturales, sociales y ambientales según sea necesario.

DI4- Valida el diseño final de acuerdo con los requerimientos, la salud y la seguridad pública, el costo total de la vida, el carbono neto cero, así como aspectos relacionados con recursos, culturales, sociales y ambientales según sea necesario.

Motivación

Los aceleradores de procesamiento de imágenes son fundamentales en sistemas embebidos, dispositivos móviles y aplicaciones de visión computacional. La interpolación bilineal es una operación típica para el *downscaling* de imágenes con calidad visual aceptable y ampliamente utilizada en GPUs, procesadores de imágenes (ISPs) y aceleradores de visión artificial.

Este proyecto permite aplicar conceptos avanzados de arquitectura de computadores en el diseño de unidades funcionales especializadas, incluyendo: paralelismo a nivel de datos mediante *SIMD*, manejo eficiente de memoria interna FPGA, aritmética de punto fijo para operaciones fraccionarias, y diseño de registros vectoriales adaptados a las dependencias de datos del algoritmo.

Mediante el diseño, implementación y verificación de una arquitectura de dominio específico [1] (DSA) basada en interpolación bilineal [2] paralela se pretende la aplicación de conocimientos de procesamiento vectorial, arquitecturas *pipelined*, y diseño de aceleradores especializados que son tendencias actuales en la industria de semiconductores.

Características generales del proyecto

1. Implementar un acelerador DSA para interpolación bilineal en imágenes en escala de grises (8 bits por píxel).
2. Tamaño máximo de imagen de entrada: 512×512 píxeles (262,144 bytes) almacenados en memoria interna FPGA (M10K/BRAM).
3. Implementar un registro configurable para factor de escala en rango de 0.5 a 1.0 en pasos de 0.05, que determina la resolución de salida proporcional.
4. Soportar dos modos operativos: Modo secuencial (1 píxel/ciclo) y Modo *SIMD* paralelo (N píxeles/ciclo, N≥4).
5. Permitir control y comunicación con PC vía JTAG o UART, recomendación usar como referencia el código de: https://github.com/Abner2111/GuiaJtag
6. Desarrollar un modelo de referencia en C/C++ con el mismo formato numérico para validación bit a bit.
7. Almacenar imagen de salida en memoria FPGA para recuperación y comparación.
8. Implementar mecanismo de ejecución paso a paso (*stepping*) para depuración.
9. Proveer registros de control y estado para configuración y monitoreo.
10. Diseñar arquitectura modular y *pipelined* [5], sintetizable para Cyclone V en DE1-SoC MTL2.
11. Definir y ejecutar un plan de validación con simulaciones y pruebas en hardware.
12. Implementar *performance counters* para FLOPs, accesos a memoria e intensidad aritmética.

Requisitos de Arquitectura

Registros de control y estado

El sistema de poseer registros de control y estado que incluyan como mínimo:

a- Registro de configuración de tamaño de imagen de entrada (ancho y alto).
b- Registro de factor de escala (formato de punto fijo o entero que represente el factor).
c- Registro de selección de modo de operación (secuencial vs. *SIMD*, y ancho *SIMD* N).
d- Registros de estado (*busy/ready, progress, errors*).
e- Registros tipo *performance counter* (FLOPs, lecturas de memoria, escrituras de memoria).
f- Registros *SIMD* para el cálculo de la interpolación bilineal

Nota: Los registros *SIMD* deben diseñarse como arreglos empaquetados de elementos de 8 bits (píxeles) que permitan:

a- Carga simultánea de múltiples píxeles desde memoria.

b- Almacenamiento temporal de grupos de píxeles vecinos para cada cálculo de salida.
c- Reutilización eficiente de píxeles entre cálculos de salida adyacentes (minimizando lecturas redundantes).

Se recomienda definir entre 4 y 8 *SIMD registers* para *buffering* de píxeles de entrada, resultados intermedios (productos ponderados) y píxeles de salida. El tamaño de cada registro *SIMD* depende del ancho de vector *SIMD* seleccionado; por ejemplo, para N=4, se requieren registros capaces de almacenar 16 píxeles (4 salidas × 4 vecinos cada uno).

Representación numérica y soporte de operaciones

Los píxeles se almacenan en memoria como enteros sin signo de 8 bits (valores 0–255). Los cálculos internos de interpolación (multiplicaciones por coeficientes fraccionarios y sumas) deben realizarse en formato de punto fijo [3]. Se recomienda formato Q8.8 (16 bits totales: 8 bits para la parte entera, 8 bits para la parte fraccionaria).

Contexto: La interpolación bilineal requiere multiplicar píxeles por pesos fraccionarios (coeficientes entre 0.0 y 1.0). Utilizar aritmética de punto fijo permite:

a. Representar fracciones con precisión controlada sin usar punto flotante.
b. Aprovechar eficientemente los bloques DSP de la FPGA Cyclone V.
c. Reducir el consumo recursos y aumentar frecuencia de operación comparado con punto flotante.
d. Mantener suficiente precisión para aplicaciones de procesamiento de imágenes.

El formato de punto fijo elegido debe ser consistente en toda la implementación: hardware (*SystemVerilog*), modelo de referencia (C/C++) y *testbenches*, para garantizar verificación bit a bit.

Requisitos de hardware

1. Plataforma objetivo: Tarjeta **DE1-SoC-MTL2** que contiene el FPGA Cyclone V.

2. Memoria: Es permitido, aunque no recomendado, el uso de memoria interna FPGA (M10K/BRAM) para almacenar imágenes de entrada y salida. La capacidad debe soportar al menos una imagen de entrada de 512×512 píxeles y una imagen de salida de tamaño correspondiente al factor de escala aplicado.

4. Implementación: Todo el diseño debe estar implementado en *SystemVerilog*, ser completamente sintetizable y soportar ejecución paso a paso para *debugging*.

5. Interfaz de comunicación: Soportar comunicación FPGA-PC mediante JTAG (recomendado usar GuiaJtag [4] como referencia) o UART para: configuración de parámetros, carga de imagen, inicio de procesamiento, lectura de imagen de salida, control de *stepping*, lectura de *performance counters*.

6. Mecanismo de *stepping*: Implementar capacidad de ejecutar operaciones paso a paso, permitiendo observar estados intermedios de registros, memoria y señales de control.

7. Registros accesibles: Todos los registros de control, estado y *performance counters* deben ser accesibles desde el host mediante la interfaz de comunicación.

Requisitos de software

1. Modelo de referencia en C/C++: Implementar la misma interpolación bilineal usando el mismo formato de punto fijo (Q8.8 u otro elegido). Garantizar que los resultados sean bit a bit idénticos al hardware para validación. Soportar las mismas configuraciones de tamaño de imagen y factor de escala.
2. Aplicación de comunicación con la PC: Desarrollar una aplicación simple (línea de comandos o GUI básica) para configurar parámetros, cargar imagen de entrada, iniciar procesamiento, recuperar imagen de salida, visualizar resultados y compararlos con el modelo de referencia, leer y mostrar *performance counters*.
3. *Testbenches* para verificación: Implementar *testbenches* de *SystemVerilog* para pruebas unitarias de módulos individuales y *testbench* de integración para el sistema completo. Cada prueba debe indicar automáticamente si pasa o falla (no se permite inspección visual como método de verificación). Validar resultados contra el modelo de referencia en C/C++.

Requisitos de documentación

Se debe entregar un artículo científico conteniendo como mínimo los siguientes aspectos:

1. Motivación y contexto del problema de reducción de imágenes.
2. Identificación de necesidades y requerimientos (funcionales y no funcionales) considerando eficiencia energética, uso eficiente de recursos FPGA, rendimiento (*throughput*), y balance entre complejidad y funcionalidad.
3. Valoración de alternativas de solución: Justificación de la arquitectura *SIMD* seleccionada vs. otras alternativas, del formato de punto fijo elegido (Q8.8 u otro), del diseño de *SIMD registers* según las dependencias de datos de la interpolación bilineal (4 píxeles por salida). Deben considerar eficiencia energética, uso eficiente de recursos FPGA, rendimiento (*throughput*), y balance entre complejidad y funcionalidad en la valoración.
4. Diseño creativo de la solución: descripción detallada de la arquitectura modular y pipeline, diagramas de bloques del sistema completo y de módulos principales, descripción del *datapath* aritmético, explicación del diseño de *SIMD registers*, descripción del flujo de control (mediante el uso de FSM por ejemplo), descripción de la interfaz de comunicación y manejo de memoria.
5. Validación del diseño: Resultados claves del sistema en simulaciones y ejecución real funcional.
6. Análisis de resultados: reporte de *performance counters*, comparación cuantitativa de rendimiento entre modo secuencial y *SIMD*, análisis de utilización de recursos FPGA.
7. Conclusiones y recomendaciones.
8. Referencias bibliográficas.

Adicionalmente, se debe entregar un plan de verificación que incluya: descripción detallada de las pruebas, casos de prueba específicos, reporte de simulaciones con resultados claros.

Entregables

Se debe entregar los siguientes archivos en un zip file mediante el TEC digital:

1. Código *SystemVerilog* sintetizable (todos los módulos del diseño).
2. Modelo de referencia en C/C++ y aplicación para comunicación con la PC.
3. Código de *testbenches* para simulaciones unitarias y de integración.
4. Plan de verificación con descripción de pruebas y reporte de resultados.
5. Artículo científico según requisitos de documentación.
6. *Bitstream* para DE1-SoC MTL2.
7. README con instrucciones de compilación, síntesis, programación de FPGA y ejecución del sistema.

No incluya ejecutables ni proyectos de quartus compilados de ser así tendrá una nota de cero.

Evaluación

El proyecto se desarrollará en grupos de 3 integrantes como máximo. La evaluación del proyecto se da bajo los siguientes rubros:

1. **Entrega de avances según la siguiente tabla (20%):**

Tabla 1. Cronograma de avances

Semana	**Actividad Principal**	**Detalle / Tareas Específicas**	**Fecha entrega**	**Comentarios y contenidos del avance**
1	Planificación y definición inicial	Revisión de especificación, referencias sobre interpolación bilineal y *SIMD*, diagrama de bloque primer nivel del sistema	Grupo2:30/10/2025 Grupo1:31/10/2025	Objetivos específicos, roles, extracción de requisitos, referencias bibliográficas, algoritmo de alto nivel de *downscaling* (sugerencia use python)
2	Diseño de arquitectura	Diagramas de bloques, *SIMD registers*, formato punto fijo	Grupo 2:06/11/2025, Grupo 1:07/11/2025	Definición completa del modelo, diagramas, justificaciones de diseño
3	Implementación base (secuencial)	Modo secuencial (1píxel/ciclo), manejo de memoria	Grupo 2:13/11/2025, Grupo 1:14/11/2025	Funciones básicas de interpolación, *testbench* unitario funcional
4	Integración *SIMD*	Modo *SIMD* (N píxeles/ciclo), *SIMD registers*, FSM de control	Grupo 2:20/11/2025, Grupo 1:21/11/2025	Validación paralela, comparación rendimiento secuencial vs *SIMD*
5	Comunicación y validación	Interfaz JTAG/UART,integración PC, validación bit a bit	Grupo 2:27/11/2025, Grupo 1:28/11/2025	Comunicación hardware-software, *stepping* funcional
6	Pruebas finales	Pruebas exhaustivas, *performance counters*, documentación	Grupo 1 y 2: 04/12/2025	Entrega completa, análisis de resultados, presentación funcional

Es obligatorio entregar todos los avances; de no entregarlos no se procede a la revisión del proyecto.

2. **Presentación Funcional (40%):**

Todos los miembros del grupo deben estar presentes en una sesión demostrativa (previa cita con el profesor) de la funcionalidad del sistema en el FPGA, en la cual se realizarán preguntas sobre cualquier etapa del sistema.

Se evaluará:

a. Funcionamiento correcto en hardware DE1-SoC.
b. Demostración de ambos modos: secuencial y *SIMD*.
c. Validación bit a bit con modelo de referencia.
d. Correcta operación de interfaz de comunicación.
e. Funcionamiento de *stepping* y observación de estados internos.
f. Lectura y análisis de *performance counters*.
g. Dominio técnico del diseño por parte de todos los integrantes.

Si la presentación no funciona en FPGA se penaliza con 50% del rubro funcional, aún si funciona en simulación.

3. **Artículo científico (20%)**

Con la descripción del proceso de diseño del sistema y análisis de resultados, según los requisitos de documentación.

4. **Plan de verificación (20%)**

Con la descripción de pruebas y resultados según los requisitos de documentación.

Si tiene dudas puede contactar al profesor por medio de correo electrónico. La entrega debe ser realizada mediante el TEC Digital en la pestaña de evaluaciones. No se aceptarán entregas después de las 11:59 PM del 04 de diciembre (Grupo 1 y 2) de 2025.

Las defensas (presentaciones funcionales) serán el 04 de diciembre (Grupo 1 y 2) de 2025.

Opcional: interfaz con SDRAM externa para imágenes mayores (+10% evaluación) Tiene que ser 100% funcional.

Notas Finales

1. Durante el proceso de diseño se deben evaluar diferentes propuestas, entiéndase como proceso de diseño a los pasos, ideas y discusiones necesarias para obtener una solución dada.
2. Es obligatoria la entrega del plan de pruebas para la revisión funcional; de no ser así no se revisará el proyecto.
3. Aunque no es obligatorio, se recomienda realizar reuniones con el profesor para obtener guía y retroalimentación durante el desarrollo.
4. Es obligatorio entregar el código fuente y un README. No se aceptan binarios como entregables; de ser así tendrá una nota de cero.

Referencias

[1] John L. Hennessy y David A. Patterson. Computer Architecture: A Quantitative Approach. Elsevier, 2017.

[2] Wikipedia. Bilinear interpolation. https://en.wikipedia.org/wiki/Bilinear_interpolation visitada el 20-10-2025.

[3] Project F. Fixed Point Numbers in Verilog. https://projectf.io/posts/fixed-point-numbers-in-verilog/ visitada el 20-10-2025.

[4] GuiaJtag repository. https://github.com/Abner2111/GuiaJtag visitada el 20-10-2025.

[5] ZipCPU. Strategies for Pipelining. https://zipcpu.com/blog/2017/08/14/strategies-for-pipelining.html visitada el 20-10-2025.