IPD432: Diseño Avanzado de Sistemas Digitales
Semestre II-2021
Tarea 4

Indicaciones generales

- La tarea se puede realizar en forma individual o en grupos de dos personas (se recomienda hacerla en grupos).
- La fecha límite de entrega para la tarea (enlaces a repositorios e informe) es el 25 de Marzo de 2022 a las 23:59 hrs. al mail *gcarvajalb@gmail.com*.
- Se habilitará un post en Piazza para que puedan canalizar las consultas y puedan ir posteando resultados preliminares que permitan el intercambio de información (en forma moderada) y sugerencias.
- **En esta ocasión, dado el contexto de la tarea y dependiendo de los resultados, la evaluación puede quedar sujeta a una sesión de interrogación donde deberán demostrar dominio de la herramienta y técnicas de diseño.**

1. Objetivos.

El objetivo de esta tarea es utilizar la base conceptual y técnica adquirida durante el semestre trabajando en el diseño, descripción, implementación y análisis de arquitecturas de hardware optimizadas a bajo nivel, para explorar el uso de paradigmas modernos de diseño a alto nivel que se revisaron en los talleres finales del curso. En específico, en esta tarea usarán Vivado HLS y la integración de componentes en sistemas heterogéneos utilizando Vivado y Vitis, utilizando como *driving example* el co-procesador de vectores que ya conocen en detalle.

Más que en la complejidad del diseño, esta tarea apunta a que profundicen en algunos aspectos que ya han revisado, poniendo más foco en la rigurosidad de los resultados, análisis, documentación, y discusión presentada, considerando tanto el proceso como el resultado final. Se dejan además intencionalmente puntos abiertos, apuntando a incentivar la investigación independiente y discusión entre pares para tomar decisiones de diseño, aplicarlas, y analizarlas consecuentemente.

Los entregables de la tarea incluyen repositorios de código y un reporte escrito. Puntos a

evaluar consideran la documentación necesaria para **reproducir** los resultados reportados [1], rigurosidad de la metodología y análisis soportado con evidencia, discusión de amplio espectro sobre el estado del arte considerando la experiencia y literatura revisada, presentación del reporte escrito, etc.

2. Material de referencia

Para trabajar con Vivado HLS, se recomienda estudiar previamente el material disponible en el siguiente enlace: `https://www.xilinx.com/support/documentation-navigation/design-hubs/dh0012-vivado-high-level-synthesis-hub.html`

Para el desarrollo de sistemas SoC utilizando los chips Zynq, se recomienda revisar como punto de partida el libro The Zynq Book (`http://www.zynqbook.com/`) para entender los conceptos principales. A partir de allí, se puede buscar material más específico en diversos artículos o foros de internet (hay bastantes, pero recuerden aplicar lectura crítica). También están disponibles grabaciones de los talleres en la carpeta compartida del curso.

También revise ejemplos de repositorios como los siguientes, utilizandolos como ejemplos de entregables para las actividades indicadas a continuación:

- `https://github.com/ColdfireMC/`
- `https://github.com/alonsorb/ddr-ram-controller-mig`
- `https://github.com/jdjotad`
- `https://gitlab.com/oscar-rc/IPD432`
- `https://www.hackster.io/adam-taylor/fpga-based-edge-detection-using-hls-192ad2`
- Mas ejemplos en `https://www.hackster.io/fpga`

3. Actividades a desarrollar.

3.1. Procesador de vectores usando HLS para Nexys4 DDR (30 puntos)

Siguiendo la línea del co-procesador desarrollado en las tareas anteriores, utilice Vitis HLS para describir un processing core que soporte el cálculo de la distancia Euclideana entre vectores de 1024 elementos de 8 bits.

A partir de un código C genérico que describa la funcionalidad, utilice las opciones de HLS para realizar el proceso de *design space exploration*, probando distintas opciones de optimización mediante pragmas de paralelización para encontrar una arquitectura que proporcione

[1] `https://www.acm.org/publications/policies/artifact-review-and-badging-current`

una baja latencia[2] de procesamiento en el processing core (asumiendo que los datos ya se encuentran disponibles en la memoria). En particular, se recomienda partir probando distintas configuraciones de las directivas `LOOP_UNROLLING`, `LOOP_PIPELINE`, y `ARRAY_PARTITION`. Una vez entendiendo los pragmas anteriores, puede probar cualquier otro tipo de optimización que considere útil o necesaria. Se hace notar que, probablemente, las configuraciones extremas del tipo completamente combinacional y pipeline completo probadas en la Tarea anterior no sean las que entreguen la menor latencia cuando se consideran las restricciones físicas del dispositivo. Como target de diseño, apunte a que la latencia del nuevo core este en el mismo rango o menos que la latencia del core reportado en la Tarea 3.

Una vez familiarizado con el flujo de trabajo para Vitis HLS, integre el módulo generado para el processing core en el sistema desarrollado en las tareas previas para comunicarse con un PC externo mediante UART. En un escenario ideal, se espera que generen mediante HLS una caja negra para reemplazar el processing core antiguo con el nuevo, sin mayores cambios al resto de las componentes. En un escenario más realista, puede necesitar realizar ajustes a su sistema de comunicación para adaptarse a la nueva arquitectura generada, por lo que deberá considerar también la complejidad de estos cambios como un criterio de diseño (si maximizar la eficiencia computacional del processing core implica rediseñar todo el sistema desde cero, puede ser necesario relajar los target). Cualquiera sean las adaptaciones internas del co-procesador, esto no debe afectar la usabilidad del sistema; en otras palabras, el sistema debe poder usarse con el mismo script que entregó para la Tarea 3. Si requiere realizar cambios al script, repórtelos y justifíquelos.

Para esta actividad debe entregar un repositorio GIT que contenga todos los archivos fuente, debidamente comentados, necesarios para sintetizar, implementar, y utilizar su sistema. No incluya el archivo del proyecto de Vivado ni el .bit, solo las fuentes a agregar para generar el proyecto de forma local. Además de los códigos, el repositorio debe incluir, al menos, un readme o wiki con una breve indicación para justificar la configuración de pragmas (en las fuentes in cluya el IP ya generado con HLS), reporte de frecuencia de operación, métricas de latencia y throughput (indicando claramente como obtuvo las métricas), y uso de recursos. Además, debe incluir todas las indicaciones necesarias para reproducir en forma local su sistema y los resultados principales usando la tarjeta Nexys4 DDR. Reporte además el tiempo estimado (observado en su computador) para el proceso de síntesis hasta la generación del bitfile. Sea lo más claro posible, haciendo uso de screenshots, animaciones o videos para reducir ambiguedades (ver ejemplos). Siéntase libre de agregar toda la información adicional que considere necesaria.

En el repositorio solo debe reportar los archivos y resultados finales. Sin embargo, documente el proceso de exploración de diseño, resultados intermedios, y observaciones, ya que le servirá de insumo para el informe escrito.

Algunas recomendaciones:

- Durante el proceso de exploración de diseño, se recomienda utilizar en todo momento una *golden reference* para verificar mediante simulación que su sistema optimizado

[2]Notar que intencionalmente se omitó usar terminos como *minimizar la latencia*. Más que apuntar a lograr *el mejor diseño posible*, se busca que exploren las posibilidades, tomen una decisión razonable e informada en base a su exploración y sus criterios, y sean consecuentes con esa decisión. No hay una solución única. De hecho, será interesante comparar las distintas implementaciones y criterios.

cumple con las especificaciones funcionales.

- Después de generar los archivos de descripción de hardware para cada diseño, abra los archivos principales en un proyecto de Vivado para observar lo que infiere el *Elaborated Design* de los códigos generados automáticamente.
- Recuerde que los reportes de timing a nivel de síntesis son conservadores, y los reportes de timing pueden variar (para bien o para mal) durante la implementación que considera todas las restricciones físicas. El timing que importa es el que se obtiene para la implementación final.
- Notar que es posible explorar, visualizar, y analizar el código generado sin necesidad de empaquetarlo en un IP. Si bien no es recomendable manipular directamente los códigos generados, si puede ser una opción si encuentra que simplifica la integración en un sistema ya existente.
- Es importante que verifique antes de la entrega que el repositorio contiene toda la información necesaria para reproducir sus resultados, lo cual puede hacer solicitando a un tercero que reproduzca el procedimiento a partir de las indicaciones antes de la fecha de entrega. No se aceptarán *correcciones* posteriores a la entrega.

3.2. Sistema heterogeneo en SoC Zynq (35 puntos)

Basándose en el flujo de diseño ilustrado en el taller de SoC realizado hace unas semanas, implemente un sistema heterogeneo con soporte de hardware para el cálculo de la distancia Euclideana entre dos vectores en la tarjeta Zybo. En este caso, el procesador ARM de la Zynq-7000 (Programmable System o PS) cumple el rol de host que puede solicitar el servicio de procesamiento de vectores a un co-procesador especializado implementado en la FPGA (Programmable Logic o PL). El PS y el PL se comunican mediante un bus AXI. Siguiendo el mismo flujo de procesamiento descrito en tareas anteriores, el host debe ejecutar comandos para enviar los datos al PL, gatillar una operación, y recibir el resultado. Todas las tareas de procesamiento se implementan en el SoC, y el computador externo solo debe utilizarse para configurar, depurar, y monitorear el PS (normalmente mediante una consola conectada por UART). Por simplicidad, considere una configuración bare-metal para el PS, aunque si lo desea, puede setear un sistema con Linux.

Debe implementar y caracterizar dos versiones del sistema: una que opere con enteros de 32 bits y otra que opere con flotantes de precisión simple (también de 32 bits). Estas deben ser analizadas en forma independiente. Se recomienda partir con la versión de números enteros, y una vez teniendo experiencia con el flujo pasar a números flotantes.

Para cada versión, caracterice y compare la latencia para realizar una operación utilizando solamente software (todo ejecutado en el PS) y con soporte de hardware (el procesador delega trabajo al PL). La latencia debe medirse desde el momento en que el procesador comienza a enviar los datos hasta que tiene el resultado final (en el caso del co-procesador PL, esto incorpora las latencias de comunicación y procesamiento). Realice este análisis para vectores de distinto largo (por ejemplo, 16, 128, 256, 512, 1024; o bien hasta que se vea limitado por el uso de recursos en el PL). Realice pruebas considerando un par de miles de

operaciones con distintos valores de datos para cada dimensión, de forma de obtener muestras estadísticamente representativas para caracterizar la latencia, poniendo atención a propiedades como la varianza, valor medio, máximos y mínimos, etc. Estime además el throughput promedio. Compare lo medido con sus estimaciones, determine la eficiencia relativa de su sistema al utilizar números enteros y flotantes, caracterice los overheads, y analice y reporte sus observaciones.

Una vez se haya familiarizado con la terminología y el flujo de diseño, y haya definido procedimientos de evaluación, explore distintas configuraciones para la arquitectura del coprocesador y el esquema de comunicación PS-PL (por ejemplo, estudie y pruebe distintos modos de configuración del bus AXI, estudie los pragamas del tipo `INTERFACE` y configuraciones de `ARRAY_PARTITION`) para definir una estrategia que permita reducir la latencia de las operaciones [3].

Similar a la actividad anterior, debe entregar un repositorio GIT con todos los archivos fuente debidamente comentados (incluyendo software y hardware) e indicaciones para reproducir en forma local los resultados reportados. Dada la mayor complejidad de los pasos para implementar un sistema heterogéneo, considere presentar las indicaciones en formato tutorial para evitar ambiguedades. El repositorio debe incluir también un resumen de los resultados principales.

3.3. Reporte escrito (35 puntos)

Prepare un informe en formato paper que resuma sus principales decisiones de diseño, hallazgos fundamentales, y resultados para cada caso estudiado en las actividades anteriores como en las tareas previas. Mantenga una perspectiva de amplio espectro que permita consolidar lo aprendido a lo largo del curso, utilizando lo observado en esta tarea y a lo largo del semestre para obtener conclusiones generales sobre las posibilidades, beneficios y desventajas que ofrecen modernos de diseño de sistemas digitales.

Se enfatiza que para cada actividad verifiquen, analicen y razonen sobre los resultados obtenidos, analizando si lo que obtienen tiene sentido y como se diferencia de lo esperado, etc.

El informe debe contener al menos los siguientes puntos:

- Reporte, análisis y discusión sobre la metodología aplicada para la exploración del espacio de diseño en cada actividad previa, justificando como llegó a la implementación planteada e indicando si hay otros enfoques posibles que no alcanzó a probar o no funcionaron. Recuerde que no se apunta a una solución única, sino que argumente sus decisiones.
- Diagramas que ilustren la estructura de los cores obtenidos y permitan justificar las métricas de desempeño obtenidas. Enfóquese en diagramas simplificados y en observaciones generales, haciendo referencia al repositorio (que se evalúa a parte) para los detalles de implementación.

[3] Similar a la actividad anterior, el objetivo no es *minimizar y llegar al mejor diseño posible*, sino que defina criterios razonables y sea consecuentes con ellos.

- Reporte y análisis de los resultados y métricas relevantes, incluyendo comparación con sus sistema desarrollados en la tarea anterior (cuando corresponda) y trade-offs asociados a cada caso.
- Discusión de los principales resultados (son esperados, sorpresivos, tienen sentido, etc.), observaciones, problemas que encontró, lecciones aprendidas, etc.
- Conclusiones y recomendaciones generales, considerando una vison global del curso, no solo de la tarea. Incluya ventajas y desventajas de usar HLS versus diseño tradicional a bajo nivel, considerando los trade-offs asociados en términos de desempeño, recursos, productividad, etc.

Su reporte debe seguir una narrativa fluida y coherente, cuidando la calidad de la presentación, usando figuras y diagramas informativos y de buena calidad para apoyar su texto, incluyendo referencias a documentos externos si lo considera necesario. No incluya código en el informe a menos que lo considere necesario para ilustrar algún punto relevante (los detalles de los códigos y decisiones especificas van en el repositorio). Habrá penalización en la nota por detalles de formato y presentación (faltas de ortografía, errores gramaticales graves, texto desalineado, figuras en mala calidad, etc.).

El informe debe tener un máximo de 8 páginas, escritas en formato IEEE conference de doble columna, utilizando los templates disponibles en el siguiente link: `https://www.ieee.org/conferences_events/conferences/publishing/templates.html` (se recomienda el uso de LaTeX).