IPD432: Diseño Avanzado de Sistemas Digitales
Semestre II-2021
Tarea 3

Indicaciones generales

- La tarea se puede realizar en forma individual o en grupos de dos personas (se recomienda hacerla en grupos).
- Debe entregar un informe escrito 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).
- Si lo desea, puede entregar su informe en inglés (opcional).
- Habrá penalización en la nota por detalles de formato y presentación (faltas de ortografía, errores gramaticales graves, texto desalineado, figuras en mala calidad, etc.). Esto quedará a criterio del profesor. Sea prolijo, ya que la evaluación del informe se realizará con el mismo criterio que se evalúa un paper.
- La fecha límite de entrega para la tarea (codigos e informe) es el 30 de Diciembre de 2021 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 una *sana competencia* sobre quien logra el mejor diseño.
- **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 que adquieran experiencia práctica sobre técnicas de diseño avanzado usando HDLs para FPGAs, caracterización y análisis de medidas de desempeño (latencia y throughput) y los trade-off asociados, además de que obtengan una perspectiva de alto nivel sobre las diferencias entre los lenguajes VHDL y SystemVerilog.

Además, a parte del trabajo técnico, también se evaluará la calidad del reporte escrito en términos de formalidad, presentación, entrega de información que permita reproducir sus experimentos y resultados, y discusión acerca del proceso y resultados.

2. Actividad de diseño e implementación.

Para esta tarea deberán diseñar, describir, implementar, y caracterizar el desempeño de un coprocesador funcionalmente equivalente al de la Tarea 2, pero en este caso utilizando arquitecturas especializadas que consideran optimizaciones para distintos objetivos de implementación. Como pudieron observar en la tarea anterior, si bien algunas de las operaciones realizadas sobre los elementos de los vectores de entrada ofrecían un grado de independencia de datos que desde el punto de vista algorítmico podrían realizarse en paralelo, el uso de primitivas de memoria tradicionales del tipo BRAM con un solo bus de lectura impedía esta optimización ya que los forzaba a operar de forma secuencial sobre los elementos de los vectores. Técnicamente hablando, aún cuando contaban con los recursos de cómputo necesarios para implementar operaciones en forma paralela (como algunos de Uds. correctamente notaron), la restricción en los accesos de memoria no les permitía acceder a los datos lo suficientemente rápido para poder aprovechar estos recursos paralelos, lo cual hace la latencia y throughput queden *restringidos por memoria* [1].

Para esta tarea utilizarán la flexibilidad ofrecida por las FPGAs para aplicar un pequeño pero significativo cambio en la estructura de almacenamiento de los datos de entrada, lo cual permitirá primero *exponer el paralelismo* inherente de las operaciones sobre vectores realizadas en el coprocesador (y que algunos de Uds. ya habían identificado), y posteriormente *explotar el paralelismo* mediante arquitecturas de cómputo especializadas.

Junto a este enunciado, se entrega el archivo comprimido `lfsr_tutorial_src.zip`, el cual contiene una descripción en VHDL de un *Linear Feedback Shift Register (LFSR)*. Dentro del diseño se incluye un módulo que describe un *adder-tree* que lee datos desde estructuras del tipo *Serial-Input-Parallel-Output* (SIPO) [2]. El primer paso será estudiar este módulo para aprovechar de familiarizarse con la semántica usada en VHDL, y posteriormente aplicar ingeniería inversa al diseño para entender su arquitectura interna, sus componentes principales, y los detalles de operación (se recomienda hacer uso de las herramientas de *Elaborated Design* y simulaciones). Una vez analizado y entendido el funcionamiento de estas componentes, identifique las primitivas útiles que puede reutilizar en su coprocesador para acelerar las operaciones `sumVec`, `avgVec`, y `manDist`. Por simplicidad, puede omitir el cálculo de la distancia Euclideana en su nuevo coprocesador (aunque sigue siendo un excelente ejercicio incluirla).

Utilizando como base el diseño de su Tarea 2, aplique los cambios necesarios a su sistema para poder integrar la nueva lógica **descrita en SystemVerilog** que permite acelerar los cálculos. En el caso de las estructuras SIPO, se recomienda estudiar el uso de los *packed and*

[1] O *memory bounded*. Se espera que hayan leído sobre esto en la investigación realizada para la tarea anterior.

[2] Este código ha sido tomado de un repositorio público. La autoría y todo el crédito corresponde a D.W. Hawkins del California Institute of Technology (Caltech). Mayor información puede encontrarse en: `http://www.kiss.caltech.edu/workshops/mmic2008/presentations/hawkins.pdf`

unpacked arrays que ofrece SystemVerilog [3].

Debe implementar, evaluar, y caracterizar dos nuevas versiones del coprocesador:

(a) Una versión con *fully-combinational datapath*, es decir, el datapath para realizar las operaciones solo debe contener lógica combinacional (sin FFs en el medio).

(b) Una versión con *fully-pipelined datapath*, es decir, particionar el datapath en etapas uniformes e incorporar FFs entre etapas para aumentar la frecuencia de operación.

En ambos casos, puede considerar que los resultados de las operaciones se guardan en una memoria interna antes de ser transmitidas por la UART hacia el host. Para cada versión del procesador, primero verifique la equivalencia funcional de las nuevas versiones con las de la tarea anterior (todas deben generar los mismos resultados para un mismo set de entradas), y luego caracterize el uso de recursos, latencia y throughput para cada operación. Para mediciones de latencia y throughput, considere el tiempo desde que se gatilla la operación correspondiente hasta que el último bit del resultado se almacena en la memoria de salida. Por simplicidad, no considere en este análisis las latencias asociadas a la comunicación por el puerto UART, asumiendo que al momento de gatillar la operación los vectores de entrada ya se encuentran almacenados en los bloques de memoria [4]. Valide sus estimaciones con las observadas por medio del analizador lógico integrado (ILA).

Para determinar la máxima frecuencia de operación en cada caso, analice los reportes de timing, aplique las técnicas que estime conveniente (tanto a nivel de diseño como de directrices para la herramienta de síntesis e implementación), e incremente la frecuencia objetivo en el constraint hasta que la herramienta comience a fallar en el análisis de timing o bien la implementación se demore más de un tiempo "razonable". Analice, reporte, y discuta sus decisiones de diseño, procedimientos seguidos, y los resultados obtenidos.

Debe entregar un informe en formato paper que contenga como mínimo: (i) una descripción en bloques de la estructura del coprocesador, incluyendo un análisis detallado sobre las decisiones de diseño que tomó, y descripción de cualquier funcionalidad o característica que haya considerado necesaria y no esté explícitamente especificada en el enunciado; (ii) procedimientos seguidos para la implementación de sus diseños y la exploración del espacio de diseño que le permitió maximizar el desempeño para las distintas métricas (los procedimientos deben ser lo suficientemente detallados para que sean repetibles y verificables por un tercero); (iii) reporte y análisis de los resultados, incluyendo comparación con sus sistema desarrollados en la tarea anterior y trade-offs asociados a cada caso; (iv) discusión de los

[3] Esta es una de las nuevas y más notorias características de SystemVerilog que pueden simplificar mucho su trabajo.

[4] La limitante de la velocidad del puerto UART es una propiedad particular de la tarjeta utilizada. Para sacarle real provecho a la arquitectura optimizada del nuevo Processing Core, sería necesario portar el diseño a una tarjeta de desarrollo de mayores prestaciones con puertos de alta velocidad (PCI express, Ethernet, etc.) para la comunicación entre procesador y acelerador. En este caso, nos interesa caracterizar la capacidad de cómputo del Processing Core asumiendo que puede operar en modo *streaming*. Si el procesador principal no es capaz de pasar los datos lo suficientemente rápido, no es culpa del acelerador. De hecho, la velocidad de comunicación entre host y coprocesador suele ser un cuello de botella común en sistemas de este tipo (*communication bounded*), como debieron leer en la investigación asociada a la tarea anterior. Alcanzar el *maximum occupancy* en el coprocesador, lo que implica que en cada ciclo las unidades de cómputo estén haciendo algo útil, suele ser bastante difícil.

principales resultados, observaciones, problemas que encontró y lecciones aprendidas; (v) un anexo breve con cualquier información particular que considere necesaria para que un usuario pueda reproducir su trabajo y utilizar su sistema. 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. Puede incluir más información si lo considera necesario para mejorar la calidad de su reporte. No incluya código en el informe a menos que lo considere necesario para ilustrar algún punto relevante.

Debe entregarse además una carpeta que incluya solo los archivos fuente necesarios para implementar su diseño. Esto incluye los archivos HDL (típicamente carpeta sources de su proyecto), constraints (típicamente carpeta constraint de su proyecto), cualquier bloque de configuración de IP que haya instanciado, e indicar cualquier setting que haya usado en la herramienta de diseño. No incluya el archivo de programación .bit, ya que este se generará al momento de la revisión y se probará en la tarjeta Nexys4 DDR. Revise muy bien que está enviando los archivos correctos y suficientes para implementar su sistema. Se penalizará la nota si no se incluyen todos los archivos necesarios y también si incluye archivos innecesarios derivados del proceso de síntesis e implementación en su computador personal.

Al preparar su informe y códigos, considere que es crítico que sus resultados sean repetibles, replicables, y reproducibles. El siguiente enlace es de lectura obligatoria: `https://www.acm.org/publications/policies/artifact-review-badging`.

Ante cualquier duda, comentario, hallazgo fundamental, discusión sobre el tema de reutilización de módulos, o consulta técnica sobre warnings o problemas en su implementación, por favor reportarlo en Piazza para fomentar la discusión grupal.

Recomendaciones.

Como se ha remarcado varias veces y ya deben tenerlo interiorizado, nunca parta un diseño sin tener al menos un diagrama de bloques de los módulos a implementar. Un buen diagrama y diseño modular le permitirá maximizar la reutilización de los módulos ya implementados en la tarea anterior. Recuerde siempre la estrategia de *dividir para conquistar*.

Una parte fundamental de este trabajo es familiarizarse con la estructura de un adder-tree y las distintas formas de implementarlo. Se recomienda partir familiarizándose con esta estructura, comenzando con ejemplos pequeños e ir escalando la complejidad de a poco (por ejemplo, vectores de 8 elementos, despues de 16 elementos, y así sucesivamente). Pruebe las distintas técnicas de descripción, y haga uso de la herramienta *Elaborated Design* en Vivado para verificar que las descripciones son correctas. Solo una vez que entienda bien los conceptos y técnicas, proceda a describir un adder-tree de mayor escala y a incorporarlo en el sistema completo. Estructure el código de tal forma que pueda maximizar el uso de parámetros para un fácil escalamiento de su diseño. Partir con estructuras simples y analizarlas en forma aislada es crítico, ya que al aumentar la complejidad se hace difícil visualizar la estructura, y además los tiempos de procesamiento en la herramienta comienzan a crecer exponencialmente.

Idealmente, si aplicó buenas prácticas de diseño en la tarea anterior, la interfaz de usuario

para comunicarse desde el PC no debería requerir ningun cambio para operar con el nuevo coprocesador. Es decir, el nuevo coprocesador debería poder operar con los mismos scripts entregados en la tarea anterior. Si requiere realizar modificaciones, ya sea por claridad o porque se da cuenta de que la solución anterior de la interfaz de usuario podría mejorarse, reporte y justifique debidamente estos cambios.

El Design Space Exploration es una etapa fundamental del diseño digital, en donde se prueban distintas alternativas de implementación en busca de la que de *mejores* resultados bajo algún criterio. Por ejemplo, Vivado permite guardar configuraciones y correr varias veces el proceso de síntesis e implementación utilizando distintos settings para los procesos de síntesis e implementación [5]. Si bien puede jugar exclusivamente con la interfaz gráfica, hay también diversa información sobre el uso avanzado de Vivado en entornos profesionales que requieran hacer Design Space Exploration, incluyendo documentación oficial de Xilinx (aunque esta suele estar enterrada en diversos manuales). Para tener una primera aproximación, puede utilizar el siguiente link y hacer búsqueda de referencias adicionales: `https://hwjedi.wordpress.com/2017/01/04/vivado-non-project-mode-the-only-way-to-go-for-serious-fpga-designers/`

[5] `https://www.xilinx.com/video/hardware/creating-and-managing-runs.html`