

Sistemas Digitales  
(66.17/86.41)

Trabajo práctico 3

**Aritmética de punto flotante**



## 1. Objetivo

El presente Trabajo Práctico tiene como objetivo que el alumno aprenda a especificar, diseñar, describir una arquitectura, simular, sintetizar e implementar en FPGA algunas funciones de una unidad aritmética de punto flotante.

## 2. Especificaciones

- a) Implementar en lenguaje descriptor de hardware VHDL dos unidades de punto flotante para realizar la suma/resta y multiplicación.

*Nota: Para la multiplicación se debe utilizar la unidad implementada durante la cursada.*

- b) Implementar en lenguaje descriptor de hardware VHDL dos unidades de punto flotante para realizar la suma/resta y multiplicación, utilizando una arquitectura pipeline.

*Nota: Para la multiplicación se debe utilizar el operando \*.*

- c) Simular de forma automatizada todas las unidades aritméticas con los vectores de prueba suministrados en el campus virtual de la materia (elegir para la prueba el tamaño de datos que se desee).

- d) Sintetizar las unidades con y sin pipeline. Comparar las velocidades de operación alcanzadas.

*Nota: Para realizar la comparación entre las dos unidades de multiplicación se debe implementar una versión sin pipeline con el operando \*.*

- e) Generar un informe (no más de 5 hojas, sin contar el código) que incluya:

- Diagrama en bloques, especificando entradas y salidas de cada bloque
- Simulaciones (incluyendo algunas capturas de pantalla)
- Tabla de resumen de síntesis, detallando: slices, Flip-Flops, LUTs utilizadas, y **frecuencia máxima de reloj** a la que es operable el circuito (todos los items, salvo la frecuencia máxima de reloj, con indicación de porcentajes de utilización).
- Código fuente VHDL.

## 3. Manejo de archivos en VHDL para testbench

### 3.1. Introducción

Se explicará a continuación el manejo de archivos en VHDL orientado a hacer un testbench o banco de pruebas para simular y verificar el funcionamiento de arquitecturas de hardware. Las funciones que se detallarán forman parte de la biblioteca std.textio, por lo cual, ésta debe ser incluida en el encabezado del banco de pruebas.

```
library IEEE;  
use IEEE.std_logic_1164.all;  
use IEEE.numeric_std.all;  
use std.textio.all;
```

### 3.2. Diseño de un banco de pruebas

El objetivo al que apunta el diseño de un banco de pruebas es a la verificación exhaustiva del correcto funcionamiento de una descripción de hardware. Para ello se utilizará un archivo de texto, el cual contiene las muestras sin procesar y el resultado de las muestras procesadas. Alimentando al dispositivo bajo prueba (DUT) con estas muestras se puede verificar a la salida si el resultado que arroja el procesamiento de las mismas es consistente con el resultado presente en el citado archivo de texto.

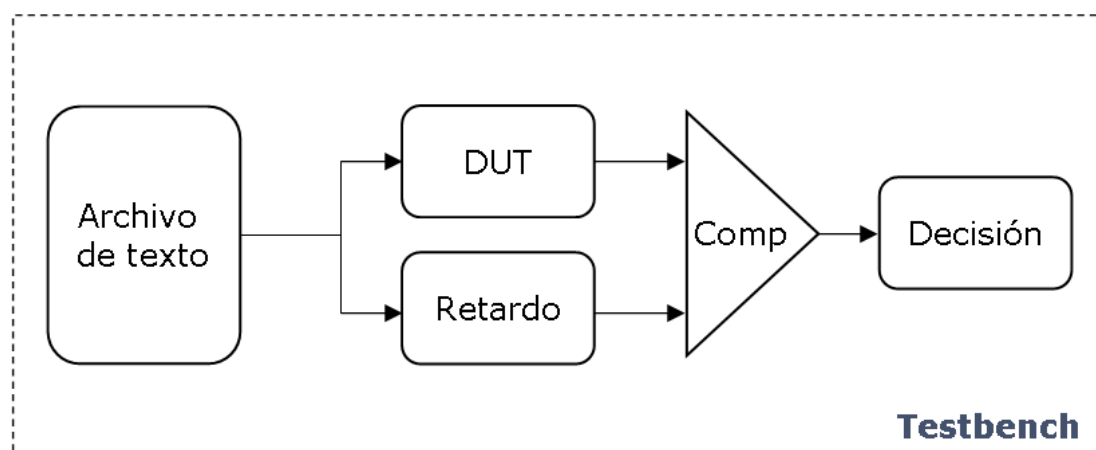


Figura 1: Diagrama en bloques del banco de pruebas

En la figura 1 se reconocen los siguientes elementos:

- **Archivo de texto:** Contiene las muestras procesadas y sin procesar. Se utilizarán las muestras procesadas como referencia para verificar el correcto funcionamiento del DUT.
- **DUT:** Device under test. Es la entidad que se desea verificar.
- **Retardo:** En ocasiones es necesario introducir un retardo o una línea de retardo a fin de compensar el retardo de procesamiento del DUT, para que el resultado que procesó el DUT se encuentre sincronizado con el resultado tomado del archivo de texto. Esta línea de retardo se puede implementar fácilmente con un shift register o registro desplazamiento.
- **Comparación:** Se comparará aquí el resultado procesado por el DUT y el resultado correcto, obtenido del archivo de texto. Para implementar la comparación y la decisión se usará un assertion statement.
- **Decisión:** De acuerdo al índice de severidad se decide qué hacer con la simulación. Se puede abortar o enviar una advertencia.

### 3.3. Lectura de archivos

Se utilizarán las funciones `getline` y `read` de la biblioteca `textio`.

La función `getline` levanta una línea de texto completa de un archivo. Dicha línea se alojará en una variable tipo `line`. Teniendo la línea de texto cargada, se procederá a leerla y recuperar los datos con la función `read`. Este procedimiento de levantar una línea de texto y leer de ella se repetirá indefinidamente hasta que termine el archivo de texto y es por ello que se debe realizar un bucle.

### 3.3.1. Ejemplo

En el campus de la materia se puede obtener el código para un posible banco de pruebas, y también los archivos de texto que incluyen los diferentes valores a ser probados.

## 4. Material de apoyo

Para confeccionar el trabajo práctico, además de la bibliografía de la materia y el material disponible en el campus virtual, se puede consultar:

- Apéndice H - Computer Arithmetic  
Autor: David Goldberg. Año: 2003  
Xerox Palo Alto Research Center  
Editorial: Elsevier Science USA
- Libro  
Computer Architecture: A Quantitative Approach  
Autor: John L. Hennessy y David A. Patterson. Año: 2006  
Editorial: Morgan Kaufmann.  
ISBN: 0123704901
- Libro  
VHDL: Analysis and Modeling of Digital Systems  
Autor: Zainalabedin Navabi. Año: 1997  
Editorial: McGraw-Hill Professional  
ISBN: 0070464790

## 5. Consideraciones adicionales

### 5.1. Modalidad de trabajo

Se recomienda enfáticamente realizar un *testbench* (simulación) por cada descripción de hardware a implementar.

Una vez que se haya comprobado fehacientemente que cada descripción funciona por separado, se procederá a integrarlas y realizar una simulación general, la cual abarcará todo el bloque aritmético.

### 5.2. Lógica en el camino del reloj

Bajo ningún concepto se aprobará un trabajo práctico en el cual haya lógica en el camino del reloj.

No utilizar más de un dominio de reloj. Utilizar sólo uno de los dos flancos de reloj (*falling\_edge* o *rising\_edge*) en todos los circuitos sincrónicos que se implementen.

### 5.3. Estilo de codificación

No utilizar señales de tipo bit o bit vector. Tampoco utilizar señales de tipo inout o buffer para puertos de entidades.

Indentar correctamente cada bloque.  
Utilizar mayúsculas sólo para las constantes.

#### **5.4. Forma de evaluación**

El informe del trabajo práctico se deberá entregar impreso (junto con el código VHDL) antes de la fecha límite dispuesta en el campus de la materia. El docente correrá una simulación con un juego de valores de vectores de prueba y verificará si el hardware diseñado funciona correctamente. Se evaluará individualmente al alumno con el trabajo práctico funcionando y, en base a las preguntas realizadas por el docente, se fijará una nota.