Universidad Tecnológica Nacional

Facultad Regional Buenos Aires

Departamento de Electrónica

Diseño Digital Con Lógica Programable

Curso R6575 - 2020

Guía de Ejercicios: Aritmética Básica
v1.0

Profesor: Ing. Martínez Garbino, Lucio José

1. Aritmetica basica

Para resolver los ejercicios de esta sección se utilizaran las siguientes convenciones:

- U(WL,FL) magnitud de WL bits, de los cuales IL=WL-FL bits son enteros y FL bits fraccionarios
- S(WL,FL) numero signado (complemento a 2) de WL bits, de los cuales IL=WL-FL bits son enteros y FL bits fraccionarios.

1. **Sumatoria**. Diseñar/Describir un sistema que calcula la sumatoria de 6 números enteros de N bits cada uno (piXi). El puerto de salida poSum debe tener la cantidad de bits (M) necesarios para almacenar el resultado sin pérdida de información. Realizar un diagrama en bloques del diseño donde se muestre el flujo de datos asi como los tamaños de todas las señales. Basar la descripción en dicho diagrama. Finalmente comparar con la vista RTL de VIVADO.

```
entity Sumatoria is
 Generic(N : NATURAL);
 Port(
  piX0,piX1,piX2,piX3,piX4,piX5:in STD_LOGIC_VECTOR(N-1 downto 0);
  poSum :  out STD_LOGIC_VECTOR(M-1 downto 0) ; );
end Sumatoria;
```

2. **Truncado**. Diseñar un módulo que implemente el método de redondeo por truncado. El puerto de entrada piX tiene formato S(10,5) y la salida poY formato S(7,2). Realizar un diagrama en bloques del diseño donde se muestre el flujo de datos asi como los tamaños de todas las señales. Basar la descripción en dicho diagrama. Finalmente realizar un testbench que cubra todos los casos posibles de entrada.

```
entity Truncado is
    Port(   piX   : in  STD_LOGIC_VECTOR(10-1 downto 0);
            poY   : out STD_LOGIC_VECTOR(7-1 downto 0) );
end Truncado;
```

3. **Redondeo más cercano**. Diseñar un módulo que implemente el método de redondeo al más cercano (Round To Nearest). El puerto de entrada piX tiene formato S(10,5) y la salida poY formato S(7,2). Realizar un diagrama en bloques del diseño donde se muestre el flujo de datos asi como los tamaños de todas las señales. Basar la descripción en dicho diagrama. Finalmente realizar un testbench que cubra todos los casos posibles de entrada.

```
entity RTN is
    Port(   piX   : in  STD_LOGIC_VECTOR(10-1 downto 0);
            poY   : out STD_LOGIC_VECTOR(7-1 downto 0) );
end RTN;
```

4. **Redondeo convergente**. Diseñar un módulo que implemente el método de redondeo convergente (Round To Even). El puerto de entrada piX tiene formato S(10,5) y la salida poY formato S(7,2). Realizar un diagrama en bloques del diseño donde se muestre el flujo de datos asi como los tamaños de todas las señales. Basar la descripción en dicho diagrama. Finalmente realizar un testbench que cubra todos los casos posibles de entrada.

```
entity Convergente is
    Port(   piX   : in  STD_LOGIC_VECTOR(10-1 downto 0);
            poY   : out STD_LOGIC_VECTOR(7-1 downto 0) );
end Convergente;
```

5. **Media aritmética**. Describir un módulo que calcule la media aritmética de 8 números enteros (piXi) formato S(12,6) y entregue el resultado en poMean formato S(8,2). La salida debe saturar en caso de desborde. Diseñar el sistema para minimizar el error de salida. Realizar un diagrama en bloques del diseño donde se muestre el flujo de datos asi como los tamaños de todas las señales. Basar la descripción en dicho diagrama.

```
entity Media is
 Port(
  piX0,piX1,piX2,piX3,piX4,piX5,piX6,piX7:in STD_LOGIC_VECTOR(12-1
      downto 0);
  poMean :  out STD_LOGIC_VECTOR(8-1 downto 0)  );
end Media;
```

6. **Ganancia digital**. Diseñar un módulo que implemente una ganancia digital multiplicando sus puertos de entrada piX formato S(12,6) y piG U(gWL,gFL). Debe entregar el resultado en el puerto de salida poY formato S(14,6). Dimensionar piG para que la ganancia tenga un rango desde 0.1 a 10 con pasos de 0.1 . En caso de desborde saturar. Diseñar el sistema para minimizar el error de salida. Realizar un diagrama en bloques del diseño donde se muestre el flujo de datos asi como los tamaños de todas las señales. Basar la descripción en dicho diagrama.

```
entity Ganancia is
 Port(
  piX : in  STD_LOGIC_VECTOR(12-1 downto 0);
  piG : in  STD_LOGIC_VECTOR(gWL-1 downto 0);
  poY : out STD_LOGIC_VECTOR(14-1 downto 0)  );
end Ganancia;
```

7. **Suma ponderada**. Describir un sistema que realiza una suma ponderada. Los datos a sumar asi como los coeficientes de ponderación son los puertos de entrada al sistema y poseen los formatos comentados en la entidad. El puerto de salida debe tener el número de bits (M) necesario para almacenar el resultado sin pérdida de información (full resolution). Realizar un diagrama en bloques del diseño donde se muestre el flujo de datos asi como los tamaños de todas las señales. Basar la descripción en dicho diagrama.

```
entity SumaPonderada is
  port(
    piX0 : in  std_logic_vector(8-1 downto 0) ; -- S(8,4)
    piX1 : in  std_logic_vector(8-1 downto 0) ; -- S(8,0)
    piX2 : in  std_logic_vector(8-1 downto 0) ; -- S(8,5)
    piW0 : in  std_logic_vector(4-1 downto 0) ; -- S(4,3)
    piW1 : in  std_logic_vector(4-1 downto 0) ; -- S(4,2)
    piW2 : in  std_logic_vector(4-1 downto 0) ; -- S(4,1)
    poZ  : out std_logic_vector(M-1 downto 0) ;
);
end SumaPonderada;
```