Módulo **`fft_L1`**

| **Entradas:** | | **Salidas** | |
|---|---|---|---|
| dinReal | 32 bits con signo | addr_in | 8 bits sin signo |
| ready_in | Bool | addr_out | 8 bits sin signo |
| | | ready_out | bool |
| | | doutReal | 32 bits con signo |
| | | doutImg | 32 bits con signo |

**Explicación básica:**

- Debe esperar a recibir **`ready_in`** antes de poder procesar 256 muestras. Después volverá a esperar por **`ready_in`** para procesar otras 256, y así sucesivamente
- Para leer un dato de memoria, ponemos la dirección de lectura en **`addr_in`** y recogemos el dato en **`dinReal`**
- Para escribir un dato en memoria, ponemos la dirección de escritura en **`addr_out`** y el dato en **`doutReal`** y **`doutImg`**

**Funcionalidad:**

Para procesar las 256 muestras, implementar el siguiente pseudocódigo:

```
for(i=0; i<256; i+=4){
       dir0 = i
       dir1 = i+2

       lectura de memoria   real0 <- [dir0]
       lectura de memoria   real1 <- [dir1]

       calcula              resReal0 = real0 + real1
                            resImg0 = 0
       calcula              resReal1 = real0 - real1
                            resImg1 = 0

       escritura en memoria [dir0] <- (resReal0, resImg0)
       escritura en memoria [dir1] <- (resReal1, resImg1)
       ____________________________________________________

       dir0 = i+1
       dir1 = i+3

       lectura de memoria   real0 <- [dir0]
       lectura de memoria   real1 <- [dir1]

       calcula              resReal0 = real0     // optimizables
                            resImg0 = -real1     // por su sencillez
       calcula              resReal1 = real0
                            resImg1 = real1

       escritura en memoria [dir0] <- (resReal0, resImg0)
       escritura en memoria [dir1] <- (resReal1, resImg1)

}
repite indefinidamente
```

Naturalmente, se deben declarar todas las variables auxiliares necesarias

Se han reutilizado varias variables, lo que puede ser una práctica cuestionable. Estaría bien que resolvieseis ese defecto en vuestra implementación.

Podéis optimizar el código siempre que siga funcionando, claro. Una posibilidad es hacer un bucle con un paso de 2 índices, en lugar de 4. Dentro del bucle habría 2 casos, el anterior, y el posterior a la línea horizontal

**Test bench:**

Hay datos para 2 ejecuciones completas consecutivas (con 256 muestras cada una)

**Entradas**: testL1_in.txt: 512 líneas  2 columnas cada una. No hay valor de ready_in, debes ponerlo tú para que empiece

Significado de las columnas:

1ª:  primer dato (dinReal) que nos pide para esa iteración

2ª:  segundo dato (dinReal) que nos pide para esa iteración

**Salidas**: testL1_out.txt: 512 líneas  8 columnas cada una. No hay valor de ready_out, el circuito lo producirá, comprueba tú mismo que es así

Significado de las columnas:

1ª: primer valor de addr_in para esa iteración

2ª: segundo valor de addr_in para esa iteración

3ª: primer valor de addr_out para esta iteración

4ª: parte real del primer resultado de esta iteración ( doutReal )

5ª: parte imaginaria del primer resultado de esta iteración ( doutImg )

6ª: segundo valor de addr_out para esta iteración

7ª: parte real del segundo resultado de esta iteración ( doutReal )

8ª: parte imaginaria del segundo resultado de esta iteración ( doutImg )