1. METODOLOGÍA Y HERRAMIENTAS DE DISEÑO DE COMPUTADORES

[Transparencia relacionada con el problema]

1.1. [*Unidades de diseño*, pág. 7].

Añade a la siguiente declaración de entidad una cláusula que defina las constantes genéricas `Tpw_clk_h` y `Tpw_clk_l` del tipo *`delay_length`* con un valor por defecto de 3 ns.

```
entity flipflop is
     port (clk, d: in bit; q, q_n: out bit);
end entity flipflop;
```

1.2. [*Unidades de diseño*, pág. 16].

Escribe una cláusula de contexto que haga accesibles las bibliotecas `company_lib` y `proyect_lib` y que haga directamente visibles las entidades `in_pad` y `out_pad` de `company_lib` y todas las entidades de `proyect_lib`.

1.3. [*Unidades de diseño*, pág. 16].

Escribe una cláusula de contexto que haga accesibles la biblioteca `DSP_lib` y que haga directamente visibles la entidad `systolic_FFT` y todos los ítems declarados en un paquete `DSP_types` (la entidad y el paquete pertenecen a la biblioteca `DSP_lib`).

1.4. [*Elementos básicos del lenguaje*, págs. 3, 9].

Escribe los siguientes literales decimales como literales hexadecimales.

`1  34  256.0  0.5`

1.5. [*Elementos básicos del lenguaje*, págs. 3, 10].

¿Cuál es la diferencia entre los literales `16#23DF#` y `X"23DF"`?

1.6. [*Elementos básicos del lenguaje*, pág. 12].

Escribe declaraciones de constante para el número de bits en una palabra de 32 bits y para el número π (3.14159).

1.7. [*Elementos básicos del lenguaje*, págs. 22, 24].

¿A qué valor se inicializa por defecto una señal o una variable de cada uno de los siguientes tipos? `std_ulogic`, `character`, `boolean`, `bit`.

1.8. [*VHDL Quick Start*, algoritmo (ciclo) de simulación; *Sentencias*, págs. 7 y 8].

En el momento actual, la señal `s` tiene el valor '0'. ¿Cuál es el valor de las variables booleanas `v1` y `v2` después de ejecutar las siguientes sentencias dentro de un proceso?

```
s <= '1';
v1 := s='1';
wait on s;
v2 := s='1';
```

1.9. [*VHDL Quick Start*, algoritmo (ciclo) de simulación].

Dada la siguiente arquitectura

```
architecture dataflow of ejemplo is
     signal a, b, c: bit := 0;
begin
     a <= '1' after 3 ns;
     b <= not a after 2 ns;
     c <= not b after 2 ns;
end dataflow;
```

A) Escribe el proceso al que equivale la asignación de señal

```
b <= not a after 2 ns;
```

B) ¿Cambiaría la arquitectura si cambiásemos el orden de las tres asignaciones de señal?

C) Completa la siguiente tabla

Ciclo de simulación	Tiempo de simulación (ns)	Valores de las señales	Transacciones programadas (señal, valor, tiempo)
		a b c	
Inicial			
1º.			
2º.			
3º.			
4º.			
5º.			

1.10. [*VHDL Quick Start,* pág. 37 (pág. 33 del fichero) y *Elementos básicos del lenguaje*, pág. 42].

A) Indica las transacciones (cada transacción es un par [valor, tiempo]) aplicadas a la señal `s` por el siguiente proceso. ¿En qué instantes hay un evento en la señal `s`?

```
process TES is
begin
     s <= 'Z', '0' after 10 ns, '1' after 30 ns;
     wait for 50 ns;
     s <= '1'after 5 ns, 'H' after 15 ns;
     wait for 50 ns;
     s <= 'Z';
     wait;
end process;
```

B) Indica los valores que van tomando a lo largo del tiempo de simulación `s'stable(5 ns)`, `s'quiet(5 ns)` y `s'transaction`. ¿Cuáles son los valores de `s'event` y `s'last_event` en el tiempo de simulación 61 ns?

1.11. [*Elementos básicos del lenguaje*, pág. 42; *Sentencias*, pág. 50].

Escribe una sentencia `assert` concurrente para verificar que el tiempo entre cambios de una señal de reloj, `clk`, es, al menos, `T_pw_clk`.

1.12. [*Sentencias*, págs. 7 y 8].

A) Escribe una sentencia `wait` que suspende un proceso hasta que una señal `s` cambia de '1' a '0' y una señal de habilitación, `en`, es '1'.

B) Escribe una sentencia `wait` que suspende un proceso hasta que una señal `ready` vale '1' o transcurren 5 ms.

1.13. [*VHDL Quick Start,* pág. 27 y ss.]

Se tiene la siguiente interfaz de un semisumador

```
entity HalfAdder is
     port(I1, I2: in bit; Cout, S: out bit);
end HalfAdder;
```

y una arquitectura para el mismo. Un banco de pruebas para esa entidad sería

```
entity TestHalfAdder is
end TestHalfAdder;

architecture Funcional of TestHalfAdder is

component HalfAdder
port(I1, I2: in bit; Cout, S: out bit);
end component;

signal A, B, COut, Sum: bit;

begin
  HA1: HalfAdder port map(A, B, COut, Sum);

  A <= not(A) after 5 ns;
  B <= not(B) after 10 ns;

  wait;
end Funcional;
```

Escribe un banco de pruebas para un sumador completo cuya declaración de entidad sea

```
entity FullAdder is
     port(A, B, Cin: in bit; Cout, Sum: out bit);
end FullAdder;
```