Fundamentos de Microprocesadores. UNIDAD 1
DISEÑO DIGITAL Y VHDL

1.1. Dadas las señales fuente y dest, ambas de tipo std_logic_vector(7 downto 0), escriba el código para que dest sea fuente desplazada aritméticamente una posición a la derecha.

1.2. Escriba el código equivalente al proceso adjunto, utilizando una sentencia concurrente de tipo *when – else*.

```
process(all)
begin
   if a < b then
      s <= b;
   elsif a < c then
      s <= c;
   else
      s <= a;
   end if;
end process;
```

1.3. Siendo *fuente* un *std_logic_vector(3 downto 0)* y *dest* un *std_logic_vector(6 downto 0)*, escriba el código para que *dest* sea igual a *fuente* multiplicado por 8 sin utilizar la multiplicación ni la suma.

1.4. Siendo *clk, reset, d* y *q* señales de tipo *std_logic*, escriba el código para que *q* sea un biestable de tipo D con reset asíncrono activo a nivel bajo y activo por flanco de subida.

1.5. Escriba una sentencia de tipo *assert* para comprobar en un test-bench que la señal s está a '1' o si no indicar "La señal s no vale 1" y parar la simulación.

1.6. Escriba el código equivalente al indicado con una única sentencia de tipo *if*. Añada un proceso si fuera necesario.

```
z <=  a when s = "00" else
      b when s = "01" else
      c;
```

1.7. Siendo *clk, reset, d* y *q* señales de tipo *std_logic*, escriba el código para que *q* sea un biestable de tipo D con reset síncrono activo a nivel bajo y activo por flanco de subida.

1.8. Escriba el código VHDL para generar un multiplexor cuya salida es la señal *z*, y cuyas entradas de datos son *a0*, *a1*, *a2* y *a3*, todas de tipo *std_logic_vector(31 downto 0)*. La señal de control es *sel*, de tipo *std_logic_vector(1 downto 0).* No hace falta que incluya la *entity* ni la *architecture*, sólo el código para la funcionalidad de la ALU (con process si es necesario).

1.9. Dada la señal *dato4*, de tipo *std_logic_vector(3 downto 0)*, escriba el código VHDL para que *salida8*, de tipo *std_logic_vector(7 downto 0)*, sea *dato4* extendida en signo.

1.10. Escriba el código equivalente al indicado con una sentencia de tipo *case*. Añada un proceso si fuera necesario.

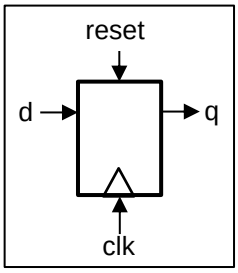
```
with s select
      z <= a when "00",
           b when "01",
           c when others;
```

1.11. Siendo *clk, reset, d* y *q* señales de tipo *std_logic*, escriba el código para que *q* sea un biestable de tipo D activo por flanco de bajada con una señal de reset asíncrono activa a nivel alto.

1.12. Escriba un proceso que genere una señal de reloj *clk* con período 20 ns (10 ns a nivel bajo y 10 ns a nivel alto), y que lo haga indefinidamente.

1.13. Escriba el código VHDL para generar una pequeña ALU con señal de selección *s*, que si está a '0' genera la suma y si está a '1' genera la and. Las entradas de datos y la salida se llaman respectivamente *a*, *b* y *z*, y son todas de tipo *std_logic_vector(31 downto 0)*. No hace falta que incluya la *entity* ni la *architecture*, sólo el código para la funcionalidad de la ALU (con process si es necesario).

1.14. Utilizando VHDL, diseñe un proceso que modele un flip-flop tipo D con entrada síncrona de Reset activo en alto y reloj activo por flanco de subida. Se adjunta un esquema de la entidad a modelar.



1.15. Escriba en la parte derecha el código equivalente al indicado con una sentencia de tipo *if*. Añada un proceso si es necesario.

```
with s select
z <=  a when "00",
      b when "01",

      c when others;
```

1.16. Diseñe un multiplexor 4 a 1 con entradas de 8 bits de las siguientes cuatro formas:
- **a.** Con una sentencia ***if***.
- **b.** Con una sentencia ***case***.
- **c.** Con una sentencia ***when else***.
- **d.** Con una sentencia ***with select***.

1.17. Diseñe un *testbench* para un multiplexor 4 a 1 con entradas de 8 bits del ejercicio anterior.

1.18. Diseñe un contador ascendente de 8 bits con reset asíncrono y carga en paralelo.

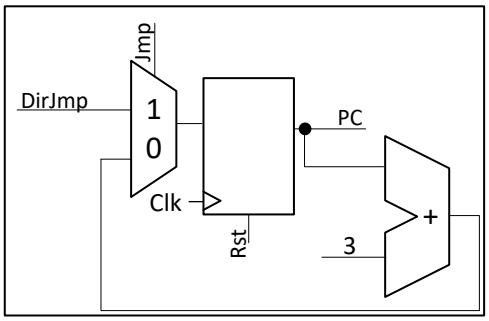
1.19. Diseñe un contador ascendente de 8 bits con reset asíncrono y carga en paralelo que se elige entre cuatro posibles. Utilice para ello los módulos diseñados anteriormente (multiplexor y contador).

1.20. Complete el siguiente bucle perteneciente a un testbench que prueba un contador descendente de 4 bits con salida llamada Q. El testbench debe generar una notificación si algún caso no es correcto. Considera que el contador inicialmente tiene el valor de 15 y que el testbench no debe comprobar desbordamientos.

```
for i in [                    ]

    assert [                                ]
    wait until Clk = '1';
    wait for 1 ns;
end loop;
```

1.21. Se desea diseñar el módulo del PC (Program Counter) de un microprocesador no MIPS. Se sabe que cada instrucción ocupa 3 bytes y que se pueden realizar saltos a la dirección DirJmp cuando la señal de control Jmp es igual a '1'. El módulo tiene una señal de reset (Rst) asíncrona activa a nivel alto la cual pone el valor del PC a 0. Complete la arquitectura en el código adjunto:




```
entity ModuloPC is port(
   Clk, Rst, Jmp : in std_logic;
   DirJmp : in unsigned(15 downto 0);
  PC : out unsigned(15 downto 0));
end ModuloPC;

architecture Problema of ModuloPC is
…
…
end Problema;
```

1.22. Dibuje el circuito que sería sintetizado con el siguiente código VHDL. Todas las conexiones deben estar etiquetadas en el esquemático. Se sabe que A es una señal de tipo signed(7 downto 0) Además, todos los componentes deben ser identificados correctamente, bien mediante un símbolo estándar o bien explicando su funcionamiento en texto auxiliar.

```
process(all)
begin
   if rising_edge(Clk) then
      if CE = '1' then
         if X = '1' then
            Y <= resize(A,16) ;
        else
            Y <= A & "00000000"
        end if;
     end if;
  end if;
end process;
```

1.23. Dibuje el circuito que sería sintetizado con el siguiente código VHDL. Todas las conexiones deben estar etiquetadas en el esquemático. Además, todos los componentes deben ser identificados correctamente, bien mediante un símbolo estándar o bien explicando su funcionamiento en texto auxiliar.

```
Q1 <= A when Ctrl1 = '1' else  B when others;
Q2 <= C when Ctrl2 = '1' else D when others;

Problema: Componente port map(PuertoA => Q1, PuertoB => Q2, Salida => Q3);
```