Descripcion secuencia-

1) process para la sentencia if else

```
--reaset asincrono
process (<clock>,<async_reset>)
begin
  if <async_reset> = '1' then
    <statements>;
  elsif (<clock>'event and <clock> = '1')
then
    if <sync_reset> = '1' then
      <statements>;
    else
      <statements>;
    end if;
  end if;
end process;
```

```
--reaset sincrono

process (<clock>)
begin
  if (<clock>'event and <clock> = '1'>)
then
    if <reset> = '1' then
      <statements>;
    else
      <statements>;
    end if;
  end if;
end process;
```

2) FFJK descripción con process

```
entity FFJK is
   Port (
                                clock: in  STD_LOGIC;
                                J : in  STD_LOGIC;
                                K : in  STD_LOGIC;
                                Reset : in  STD_LOGIC;
                                Salida : out  STD_LOGIC);
end FFJK;

architecture Behavioral of FFJK is
        signal selector: STD_LOGIC_VECTOR (1 DOWNTO 0) ;
        signal sSalida: STD_LOGIC;
begin
        selector <= J & k;
process (clock, Reset,selector,sSalida)
begin
  if Reset='1' then
    sSalida  <= '0';
  elsif (clock'event and clock='1') then
        case (selector) is
                when "00" => sSalida <= sSalida ;
                when "01" => sSalida <= '0';
                when "10" => sSalida <= '1';
                when "11" => sSalida <= not sSalida;
                when others => sSalida <= 'Z' ;
    end case;
  end if;
        Salida<=sSalida;
end process;
end Behavioral;
```

3) Decodificador de 4 to 2

Objetivo:

- Reconocer un proceso secuencial.
- Conocer la construcción "if / else"
- Interpretar el concepto de “proceso concurrente”.
- Utilizar Selección de dispositivo (CE)
- Utilizar la herramienta de plantilla del software.
- Utilizar las salidas del tipo “1”; ”0” y ”Z”.
- Destacar la importancia de “Reset” por hardware.
- Implementar en un CPLD un decodificador de 4 to 2.

Especificaciones de entradas / salidas

pReset : in std_logic.

pInput : in std_logic_vector(3 downto 0)
pOutput : out std_logic_vector(1 downto 0)
Clk : in std_logic

Referencia

Reset	pImput				pOuput	
0	0	0	0	1	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	1	0	0	0	1	1
1	x	X	x	x	0	0

Pasos a seguir

1. Diseñar la entidad y la arquitectura.
2. Dentro del cuerpo de la arquitectura, se debe describir el comportamiento como se indica en la tabla.
3. Utilizar una arquitectura secuencia. Utilizar dentro de una proceso la construcción “when/ case".
4. Probar la correcta descripción mediante la simulación.
5. Utilizar las salidas del tipo “1”; ”0” y ”Z”.

4) Que hace esta descripcion

```
entity d2to4 is
   Port ( A : in std_logic_vector(1 downto 0);
          E : in std_logic;
          D : out std_logic_vector(3 downto 0));
end d2to4;

architecture Behavioral of d2to4 is
begin
        process (A, E)
        begin
                if (E='0') then
                        D <= "0000";
                else
                        case A is
                        when "00" => D <= "0001";
                        when "01" => D <= "0010";
                        when "10" => D <= "0100";
                        when "11" => D <= "1000";
                        when others =>D <= "zzzz";
                        end case;
                end if;
        end process;
end Behavioral;
```

5) Que hace el siguiente código

```
entity Demux_1to8 is
   Port ( pE : in std_logic;
          pSelector : in std_logic_vector(2 downto 0);
          pEntrada : in std_logic_vector(7 downto 0);
          pSalida : out std_logic);
end Demux_1to8;
```

```
architecture Demultiplexador of Demux_1to8 is
begin
process (pEntrada,pE,pSelector)
begin
if (pE='0') then
  pSalida <= '0';
else
 case pSelector is
   when "000" => pSalida <= pEntrada(0);
   when "001" => pSalida <= pEntrada(1);
   when "010" => pSalida <= pEntrada(2);
   when "011" => pSalida <= pEntrada(3);
   when "100" => pSalida <= pEntrada(4);
   when "101" => pSalida <= pEntrada(5);
   when "110" => pSalida <= pEntrada(6);
   when "111" => pSalida <= pEntrada(7);
   when others => NULL;
   when others => pSalida <='0';
  end case;
 end if;
end process;
end Demultiplexador;
```