```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;



Entity lanzarDados is
port(
    CLK,RESET: in std_logic;
                TirarDado,TirarDado2,TirarDado3 : in std_logic;
                Display : out std_logic_vector(3 downto 0);
                Salida,Salida1 : inout std_logic;
                Segmentos : out std_logic_vector(6 downto 0)
    );
END ENTITY;

Architecture Multiplexado of lanzarDados is
--SIGNAL DEL DIVISOR DE FRECUENCIA
                SIGNAL CLK_AUX : STD_LOGIC;
                SIGNAL CONTADOR : INTEGER RANGE 0 TO 2000000 := 0;
--              SIGNAL CONTADOR1 : INTEGER RANGE 0 TO 900000 := 0;
--              SIGNAL CLK_AUX1: STD_LOGIC;

--seniales para el multiplexado
constant maximo: integer := 100000;
signal resetCont: integer range 0 to maximo;
signal resetEstado: std_logic_vector(1 downto 0) := (others => '0');
signal selDisplay: std_logic_vector(3 downto 0) := (others => '0');
--maquina de estado para el primer lanzamiento de dados
type ESTADOS is (Q0,Q1,Q2,Q3,Q4,Q5,Q6);
type ESTADOS_M is (Q0,Q1,Q2,Q3);
signal estadoP,estadoSig: ESTADOS;
```

```
signal estadoP1,estadoSig1: ESTADOS;
signal estadoP2,estadoSig2: ESTADOS;
signal estadoPM, estadoSigM: ESTADOS_M;

BEGIN
Display <= selDisplay;

Divisor_Frecuencia: process(CLK)
   Begin
   if rising_edge(CLK) then
      if(CONTADOR = 2000000) then
         CLK_AUX <= NOT CLK_AUX;
         CONTADOR <= 0;  -- Reiniciar el contador
      else
         CONTADOR <= CONTADOR + 1;  -- Incrementar el contador
      end if;
   end if;
end process Divisor_Frecuencia;
--fin del proceso del divisor
              SALIDA <= CLK_AUX;

--Divisor_Frecuencia1: process(CLK)
--    Begin
--    if rising_edge(CLK) then
--       if(CONTADOR1 = 900000) then
--          CLK_AUX1 <= NOT CLK_AUX1;
--          CONTADOR1 <= 0;  -- Reiniciar el contador
--       else
--          CONTADOR1 <= CONTADOR1 + 1;  -- Incrementar el contador
--       end if;
--    end if;
```

```
--end process Divisor_Frecuencia1;
----fin del proceso del divisor
--              SALIDA1 <= CLK_AUX1;



multiplexado: process(CLK)
Begin
 if rising_edge(CLK) then
   if resetCont < maximo then
                resetCont <= resetCont + 1;
   else
                resetEstado <= resetEstado + 1;
                        resetCont <= 0;
               end if;
 end if;
 end process multiplexado;


 --proceso maquina1


 maquina1: process(Salida,RESET)
 begin
  if RESET = '1' then
    estadoP <= Q0;
                elsif rising_edge(Salida) then
                estadoP <= estadoSig;
                end if;
              end process maquina1;


 maquina2: process(Salida,RESET)
 begin
  if RESET = '1' then
```

```
    estadoP1 <= Q0;
                elsif rising_edge(Salida) then
                estadoP1 <= estadoSig1;
                end if;
               end process maquina2;

maquina3: process(Salida,RESET)
begin
 if RESET = '1' then
    estadoP2 <= Q0;
                elsif rising_edge(Salida) then
                estadoP2 <= estadoSig2;
                end if;
               end process maquina3;

maquina4: process(Salida,RESET)
begin
 if RESET = '1' then
    estadoPM <= Q0;
                elsif rising_edge(Salida) then
                estadoPM <= estadoSigM;
                end if;
               end process maquina4;




transicion: process(estadoP,TirarDado)
begin
estadoSig <= estadoP;
case estadoP is
```

```
when Q0 =>
 if TirarDado = '1' then
              estadoSig <= Q1;
                        else estadoSig <= Q0;
                        end if;
            when Q1 =>
             if TirarDado = '0' then
              estadoSig <= Q2;
                        else
                        estadoSig <=Q1;
                        end if;

 when Q2 =>
  if TirarDado = '0' then
              estadoSig <= Q3;
                        else
                        estadoSig <=Q2;
                        end if;

            when Q3 =>
             if TirarDado = '0' then
              estadoSig <= Q4;
                        else
                        estadoSig <=Q3;
                        end if;

            when Q4 =>
             if TirarDado = '0' then
              estadoSig <= Q5;
                        else
                        estadoSig <=Q4;
```

```
                    end if;


  when Q5 =>
              if TirarDado = '0' then
               estadoSig <= Q6;
                    else
                    estadoSig <=Q4;
                     end if;


            when OTHERS =>
             if TirarDado = '0' then
               estadoSig <= Q1;
                     else
                    estadoSig <= Q6;
                     end if;
            end case;
            end process  transicion;


transicion2: process(estadoP1,TirarDado2)
begin
estadoSig1 <= estadoP1;
case estadoP1 is
 when Q0 =>
  if TirarDado2 = '1' then
               estadoSig1 <= Q1;
                    else estadoSig1 <= Q0;
                    end if;
            when Q1 =>
             if TirarDado2 = '0' then
               estadoSig1 <= Q2;
                    else
```

```
                    estadoSig1 <= Q1;
                    end if;

when Q2 =>
 if TirarDado2 = '0' then
          estadoSig1 <= Q3;
                    else
                    estadoSig1 <= Q2;
                    end if;

         when Q3 =>
          if TirarDado2 = '0' then
           estadoSig1 <= Q4;
                    else
                    estadoSig1 <= Q3;
                    end if;

         when Q4 =>
          if TirarDado2 = '0' then
           estadoSig1 <= Q5;
                    else
                    estadoSig1 <= Q4;
                    end if;

 when Q5 =>
           if TirarDado2 = '0' then
            estadoSig1 <= Q6;
                     else
                    estadoSig1 <= Q5;
                     end if;
```

```
            when OTHERS =>
              if TirarDado2 = '0' then
                estadoSig1 <= Q1;
                        else
                        estadoSig1 <= Q6;
                        end if;
            end case;
            end process  transicion2;

transicion3: process(estadoP2,TirarDado3,Detener)
 begin
 estadoSig2 <= estadoP2;
 case estadoP2 is
  when Q0 =>
   if TirarDado3 = '1' then
                estadoSig2 <= Q1;
                        else estadoSig2 <= Q0;
                        end if;
            when Q1 =>
             if TirarDado3 = '0' then
              estadoSig2 <= Q2;
                        else
                        estadoSig2 <= Q1;
                        end if;

   when Q2 =>
    if TirarDado3 = '0' then
                estadoSig2 <= Q3;
                        else
                        estadoSig2 <= Q2;
                        end if;
```

```
            when Q3 =>
             if TirarDado3 = '0' then
               estadoSig2 <= Q4;
                        else
                        estadoSig2 <= Q3;
                        end if;

            when Q4 =>
             if TirarDado3 = '0' then
               estadoSig2 <= Q5;
                        else
                        estadoSig2 <= Q4;
                        end if;

when Q5 =>
             if TirarDado3 = '0' then
               estadoSig2 <= Q6;
                        else
                        estadoSig2 <= Q5;
                        end if;

            when OTHERS =>
             if TirarDado3 = '0' then
               estadoSig2 <= Q1;
                        else
                        estadoSig2 <= Q6;
                        end if;
            end case;
            end process  transicion3;
```

```
transicion4: process(estadoPM, TirarDado3)
begin
estadoSigM <= estadoPM;
case estadoPM is
when Q0 =>
  if TirarDado3 = '0' then
        estadoSigM <= Q0;
              else estadoSigM <= Q1;
      end if;
when Q1 =>
  if TirarDado3 = '0' then
        estadoSigM <= Q2;
              else
      estadoSigM <= Q1;
              end if;
when Q2 =>
 if TirarDado3 = '0' then
     estadoSigM <= Q3;
          else
         estadoSigM <= Q2;
          end if;
When others =>
  if TirarDado3 = '0' then
    estadoSigM <= Q1;
          else
         estadoSigM <= Q3;
          end if;
end case;
end process transicion4;
```

```
mostrarDisplay: process(resetEstado,selDisplay,estadoP,estadoP1,estadoP2,estadoPM)
Begin
case resetEstado is
when "00" =>
  selDisplay <= "0111"; --enciende el primer display

when "01" =>
  selDisplay <= "1011"; --enciende el segundo display

when "10" =>
  selDisplay <= "1101"; --enciende el tercer display

when others =>
  selDisplay <= "1110"; --Se prende el 4to display
end case;

        --MOSTRAR LANZAMIENTO DE DADOS CORRESPONDIENTE A CADA MAQUINA
        case selDisplay is
        when "0111" =>
          case estadoP is
                when Q0 =>
                  Segmentos <= "1111111";
                when Q1 =>
                  Segmentos <= "1001111";
                when Q2 =>
                  Segmentos <= "0010010";
                when Q3 =>
                  Segmentos <= "0000110";
                when Q4 =>
                  Segmentos <= "1001100";
                when Q5 =>
```

```
                Segmentos <= "0100100";
               when others =>
                Segmentos <= "0100000";
               end case;

when "1011" =>
 case estadoP1 is
               when Q0 =>
                Segmentos <= "1111111";
               when Q1 =>
                Segmentos <= "1001111";
               when Q2 =>
                Segmentos <= "0010010";
               when Q3 =>
                Segmentos <= "0000110";
               when Q4 =>
                Segmentos <= "1001100";
               when Q5 =>
                Segmentos <= "0100100";
               when others =>
                Segmentos <= "0100000";
               end case;

when "1101" =>
 case estadoP2 is
               when Q0 =>
                Segmentos <= "1111111";
               when Q1 =>
                Segmentos <= "1001111";
               when Q2 =>
                Segmentos <= "0010010";
```

```
                    when Q3 =>
                     Segmentos <= "0000110";
                    when Q4 =>
                     Segmentos <= "1001100";
                    when Q5 =>
                     Segmentos <= "0100100";
                    when others =>
                     Segmentos <= "0100000";
                    end case;

            when others => --proceso para el cuarto display
             case estadoPM is
                    when Q0 =>
                     Segmentos <= "1111111";
                    when Q1 =>
                     Segmentos <= "0000001";
                    when Q2 =>
                     Segmentos <= "1001111";
                    when others =>
                     Segmentos <= "0010010";
                     end case;

                    end case;
end Process mostrarDisplay;
end Architecture;
```