Práctica de Verificación

En esta parte de la asignatura el alumno debe realizar, a partir de la especificación de una máquina de vending, un testbench para verificar el diseño realizado por un compañero basándose en dicha especificación.
Previamente debe realizar la matriz de cobertura funcional correspondiente y documentar los tests a realizar.
La verificación debe ser autónoma. Es decir, la generación de estímulos y la comprobación de resultados no requieren intervención, y la simulación muestra en consola el resultado.
El diseño se asume incorrecto. El alumno debe recopilar la evidencia de cómo el test bench ha reportado error en la simulación en una primera fase, los cambios realizados en el código para resolver el problema, y el log de la simulación ya correcta.

Especificación

Puertos

Entradas:

Clk: reloj 100Mhz std_logic
Rst: reset std_logic
c: pulso para introducir moneda std_logic
a: valor de moneda introducido unsigned 8 bits
s: precio del producto unsigned 8 bits

Salidas

D: pulso de salida válida std_logic
Cambio: valor devuelto por la máquina unsigned 9 bits
d_cambio: señal de devolución std_logic
Error: señal de error std_logic

Especificaciones

1) Precio máximo 2€.

2) Acepta monedas de 5c, 10c, 20c, 50c, 1E y 2E.
3) Devuelve el cambio.
4) Número máximo de monedas en cualquier orden: 10 monedas.
   Si se superan las 10 monedas se activa la señal de error, se devuelve el dinero introducido y vuelve a reposo.
5) Si tras introducir moneda sin llegar al precio se superan 40 ciclos de reloj sin introducir moneda, se activa la señal de error y se pone el saldo a 0, reseteando el sistema.

Diseño FSM del alumno:

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity top is
    Port ( clk : in STD_LOGIC;
           rst : in STD_LOGIC;
           s : in UNSIGNED(7 downto 0);
           a : in UNSIGNED(7 downto 0);
           c : in STD_LOGIC;
           d : out STD_LOGIC;
           cambio : out UNSIGNED(8 downto 0);
           d_cambio : out STD_LOGIC;
           error : out STD_LOGIC);
end top;

architecture Behavioral of top is
    type STATE is (Init, Waitt, Add, Disp);
    signal PSTATE: STATE := Init; --Estado inicializado a Init.
    signal tot: unsigned(8 downto 0); --Suma del saldo introducido.
    signal Nmonedas : unsigned (3 downto 0) := "0000"; --Contador de monedas introducidas.
    signal cuenta_tiempo : std_logic :='0'; --Inicia cuenta de tiempo
    signal tiempo : unsigned (4 downto 0) :="00000"; --Contador del número de ciclos de inactividad.
begin
    process (rst, clk)
    begin
        if (rst= '1') then --Se resetea al estado inicial.
            PSTATE <= Init;
            tot<=(others =>'0');
            error<='0';
        elsif (clk='1' and clk'EVENT) then
            case PSTATE is
                when Init =>
                    d<='0';
                    tot<=(others =>'0');
                    error<='0';
                    d_cambio<='0';
                    cuenta_tiempo<='0';
                    tiempo<=(others => '0');
                    cambio<=(others => '0');
                    Nmonedas<=(others =>'0');
                    PSTATE <=Waitt;
                when Waitt =>
                    if cuenta_tiempo='1' then --Se incrementa el número de ciclos
                        if tiempo <30 then    --si no se han alcanzado los 30 ciclos
                            tiempo<=tiempo+1; --de inactividad.
                        else
                            error<='1';       --Si se alcanzan los 30 ciclos de
                            PSTATE<=Init;     --inactividad, se activa la señal
                        end if;               --de error y se resetea el sistema.
                    end if;
                    if Nmonedas < 10 then
                        if (c='1') then
                            if(a=5 or a=10 or a=20 or a=50 or a=100 or a=200) then
                                Nmonedas<=Nmonedas+1; --Si se introduce una moneda válida,
                                cuenta_tiempo<='1';   --se incrementa el número de monedas
                                tiempo<=(others =>'0');--introducidas,se resetea el período
                                PSTATE <= Add;         --de inactividad y activa contador de tiempo.
                            else
                                error<='1';            --Si la moneda introducida es incorrecta,
                            end if;                    --se activa la señal de error.

                        elsif (tot>=s) then --Si se introduce un saldo igual o mayor al precio
                            PSTATE <= Disp; --del refresco, la máquina dispensa.
                        end if;
                    else
                        error<='1';   --Si se introducen más de 10 monedas,
                        d_cambio<='1';--se activa la señal de error, la
                        cambio<=tot;  --máquina devuelve el saldo y se resetea
                        PSTATE<=Init; --el sistema.
                    end if;
                when Add =>
                    tot<=tot+a;        --suma saldo
                    PSTATE <= Waitt;
                when Disp =>
                    d<='1';            --Cuando la máquina dispensa el producto,
                    d_cambio<='1';     --se activa la señal de devolución del
                    cambio<=tot-s;     --cambio, la señal de dispensar el producto,
                    PSTATE<=Init;      -- el cambio y se resetea el sistema.
                end case;
        end if;
    end process;
end Behavioral;
```