Benemérita Universidad Autónoma de Puebla

Facultad de ciencias de la computación

Arquitectura de computadoras - Examen 3

Alumnos:

Amador Ortega Christian Amauri - 201927821

Bryan Arturo Cedillo García – 201934964

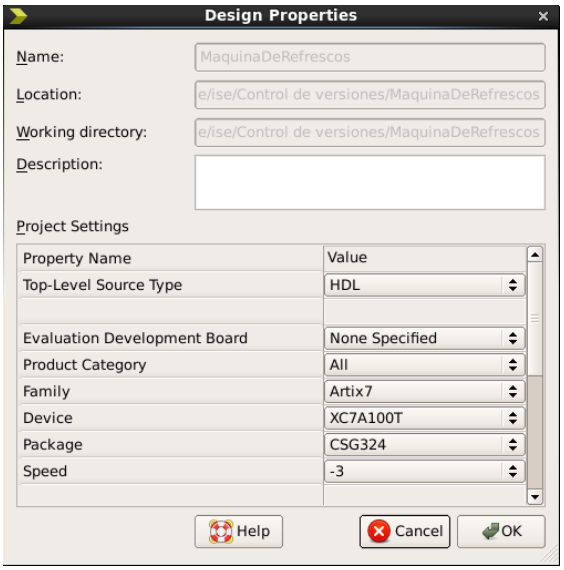
Docente: Lilia Mantilla Narváez

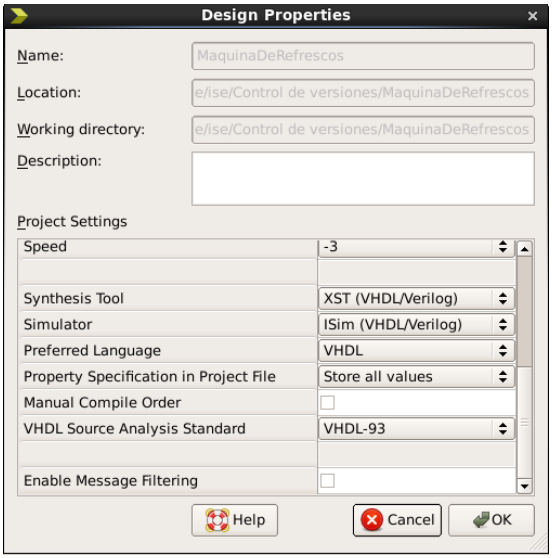
Problema:

En equipo realiza la Unidad de Control, utilizando máquinas de estado en VHDL, para controlar una máquina expendedora de refrescos. No deben diseñar los módulos de cada parte de la máquina expendedora, solo es la unidad de control.

Debe haber señales de entrada como lo son la selección del producto, el ingreso de dinero, etc. Y señales de salida como enviar cambio, soltar producto, etc.

Parámetros del proyecto:





Nota técnica:

Versión de ISE DESIGN: 14.7 (máquina virtual "Linux Oracle 64-bit" para Windows 10 y 11) (en este caso, windows 11)

Código vhdl:

- ***UnidadDeControl***

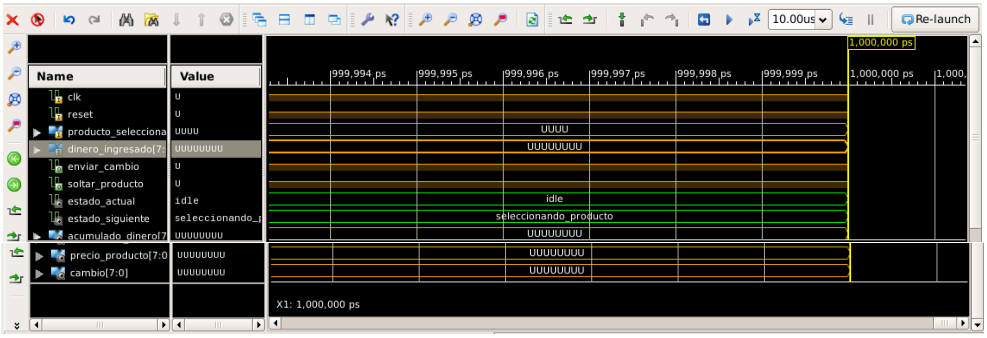
Es una descripción en VHDL de una Unidad de Control para una máquina expendedora. Utiliza una máquina de estados para controlar el flujo de operaciones. La Unidad de Control tiene diferentes estados, como "idle" (inactivo), "seleccionando_producto", "recibiendo_dinero", "dispensando_producto" y "enviando_cambio". Los estados cambian en función de las entradas del reloj, el reset, la selección del producto y el dinero ingresado. La señal "acumulado_dinero" registra la cantidad total de dinero ingresado por el usuario. Dependiendo del estado actual, se realizan las acciones correspondientes

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

entity UnidadDeControl is
    port (
        clk : in std_logic;
        reset : in std_logic;
        producto_seleccionado : in std_logic_vector(3 downto 0);
        dinero_ingresado : in std_logic_vector(7 downto 0);
        enviar_cambio : out std_logic;
        soltar_producto : out std_logic
    );
end UnidadDeControl;



architecture Behavioral of UnidadDeControl is
    -- Estados de la mquina de estado
    type estados_t is (idle, seleccionando_producto,
recibiendo_dinero, dispensando_producto, enviando_cambio);
      signal estado_actual, estado_siguiente : estados_t;
    signal acumulado_dinero : std_logic_vector(7 downto 0);
    signal precio_producto : std_logic_vector(7 downto 0);
    signal cambio : std_logic_vector(7 downto 0);




begin
    -- Proceso de la maquina de estado
    fsm_process : process(clk, reset)
    begin
             acumulado_dinero <= dinero_ingresado;
        if reset = '1' then
```

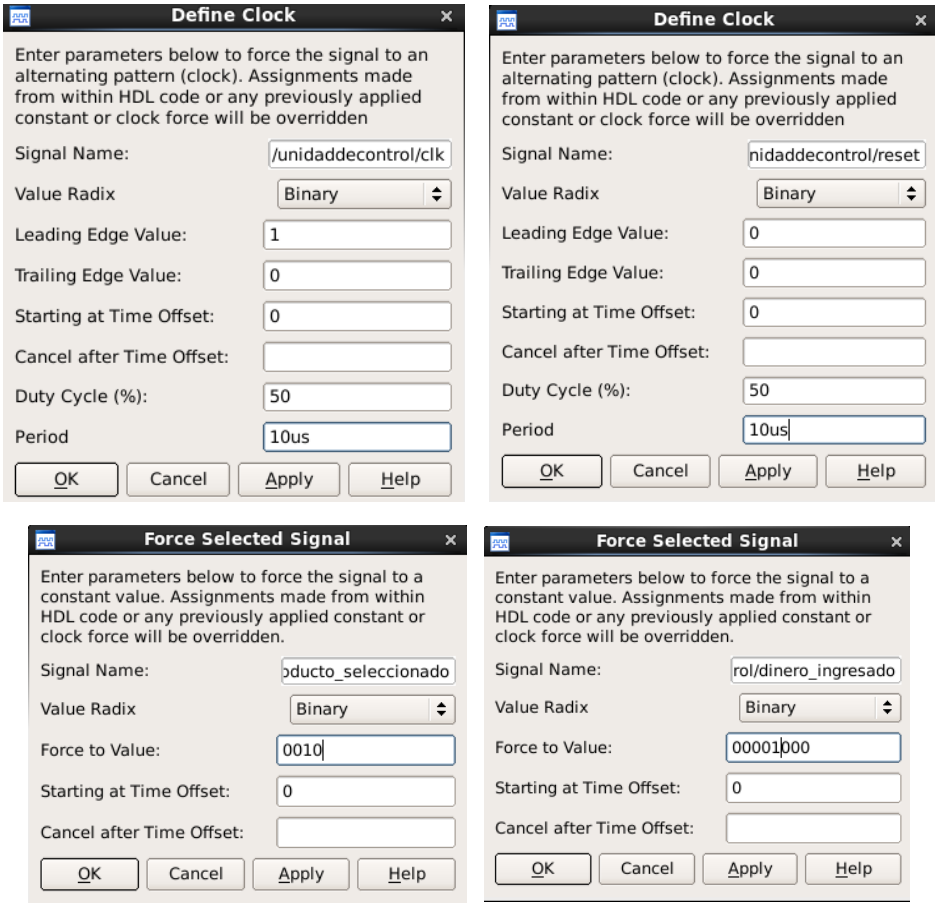
```
            estado_actual <= idle;
        elsif rising_edge(clk) then
            estado_actual <= estado_siguiente;
        end if;
    end process fsm_process;



    -- Logica de control
   control_logic : process(estado_actual, producto_seleccionado,
dinero_ingresado)
   begin

       estado_siguiente <= estado_actual;
       case estado_actual is
           when idle =>
               if producto_seleccionado /= "0000" then
                   estado_siguiente <= seleccionando_producto;
               end if;

           when seleccionando_producto =>
                if dinero_ingresado /= "00000000" then
                    estado_siguiente <= recibiendo_dinero;
                    precio_producto <= "00000100";
                end if;

           when recibiendo_dinero =>
                if acumulado_dinero >= precio_producto then
                    estado_siguiente <= dispensando_producto;
                end if;

           when dispensando_producto =>
                soltar_producto <= '1';
                estado_siguiente <= enviando_cambio;

           when enviando_cambio =>
                enviar_cambio <= '1';
                cambio <= acumulado_dinero - precio_producto;
                estado_siguiente <= idle;

           when others =>
                estado_siguiente <= idle;

        end case;
    end process control_logic;

end Behavioral;
```

Simulación:

Estado inicial:

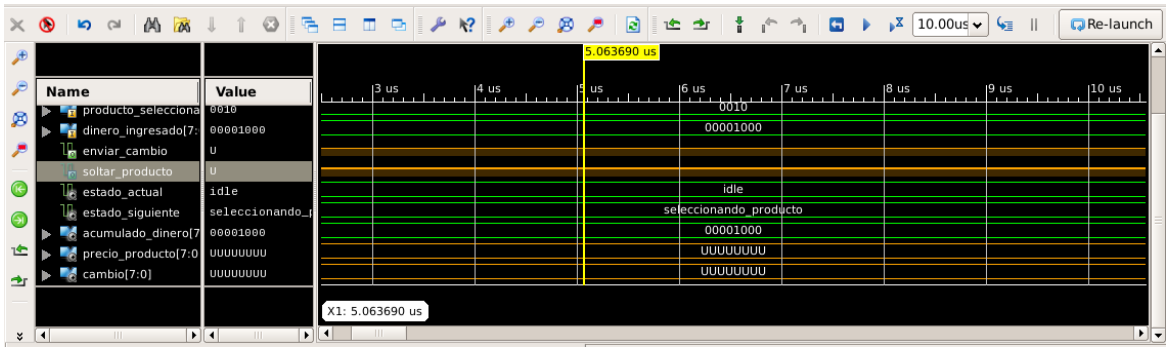


Parámetros de simulación (entradas):

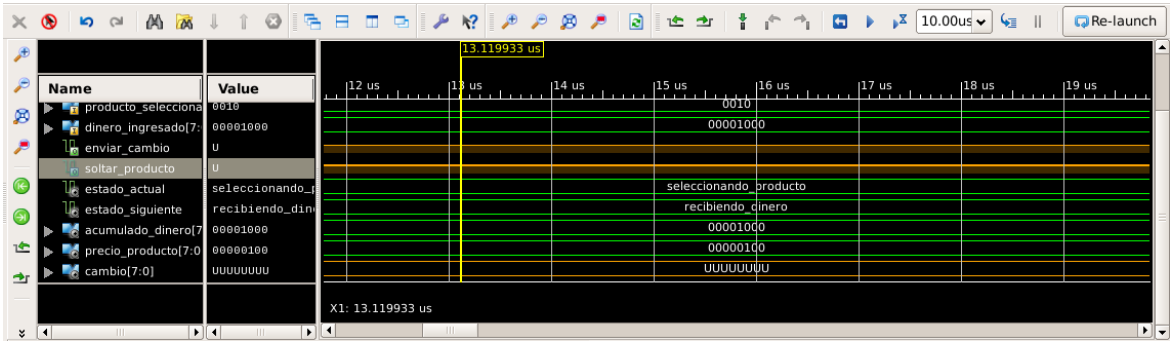


(clock: clk y reset. Constant: producto_seleccionado y dinero ingresado)

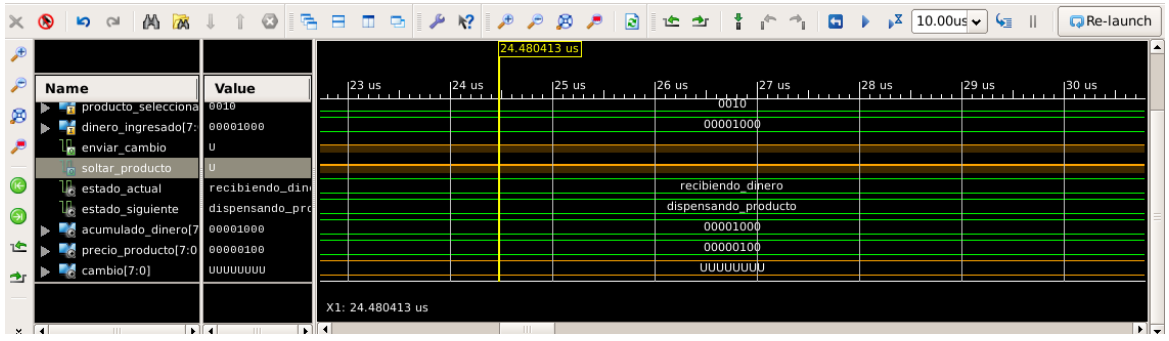
Con cada iteración, los estados correspondientes van tomando lugar, y con ellos, los valores esperados de las señales:



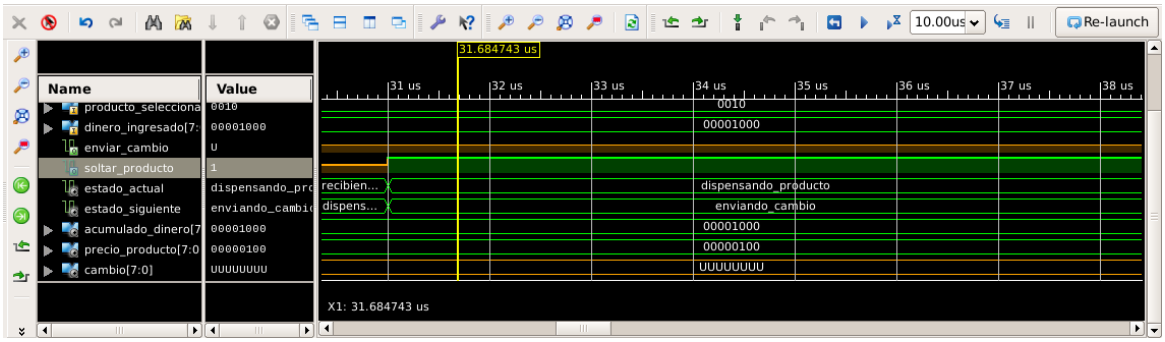
Nótense los cambios de estado…

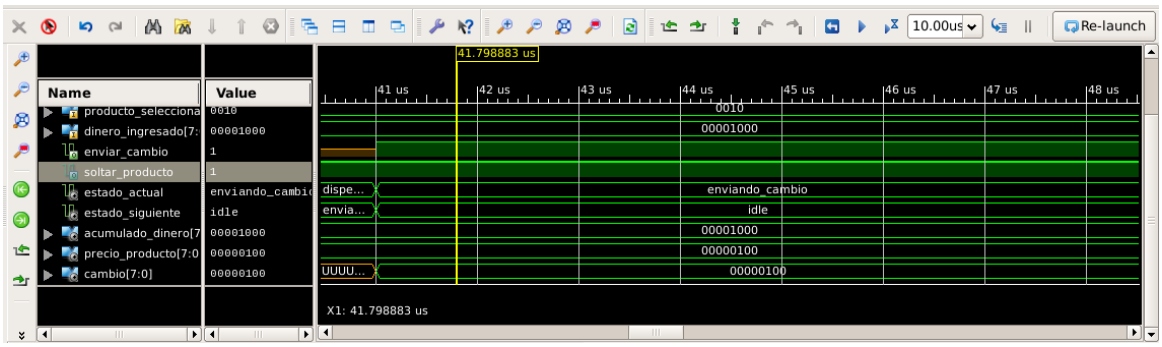


Los std_logic_vector de dinero acumulado y precio del producto toman valor…

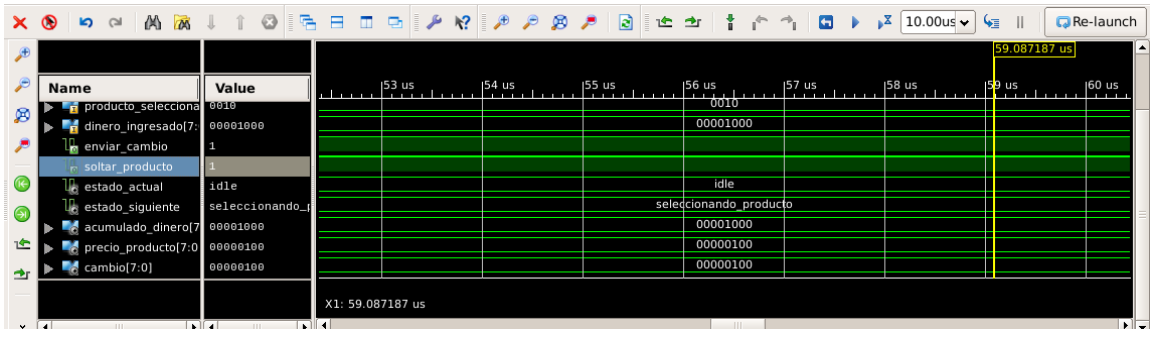


Al igual que las salidas std_logic “soltar producto” y “enviar cambio” en ese orden…





Al recibir en la entrada clk: reset un ‘1’ Los valores internos deben tomar un estado inicial definido. Para este proyecto el punto principal es demostrar el funcionamiento de la máquina de estados. Por eso no se ha incluido esa parte, mas que un regreso al estado “idle”. Además de otras partes como potenciales usos de las señales para salida como: “cambio”, “precio”, “dinero acumulado” o definir un uso interno para la señal de entrada “Producto seleccionado” (nótese que realmente esa señal no es usada). Lo que usamos son los estados.



(los valores no cambian, hay que definirlos. Pero los estados se reinician)