Práctica: Sistema automático de mezcla de leche mediante control de velocidad con FPGA.

Profesor: Eric Ramos Aguilar

I. Objetivo.

Desarrollar y simular en VHDL un sistema de control automatizado para el llenado de un contenedor de leche utilizando una FPGA, para gestionar el encendido y apagado de un motor con una hélice para el mezclado que opere a diferentes velocidades según el nivel de llenado del contenedor, que será medido mediante sensores.

II. Desarrollo.

El sistema debe controlar un motor de mezcla que opera a diferentes velocidades (tres velocidades) y activar sensores en tiempos específicos para detectar el nivel de llenado, estos son:

- Sensor de nivel de llenado bajo: 20 segundos → Velocidad 1 a los 22 segundos.
- Sensor de nivel de llenado medio: 45 segundos → Velocidad 2 a los 47 segundos.
- Sensor de nivel de llenado máximo: 70 segundos → Velocidad 3 a los 72 segundos.

II-A. Definición de entradas y salidas.

Componente	Entrada/Salida	Descripción
Válvula	SW7-N3	Activa/desactiva el sistema (reset)
LED de válvula activada	LD-7	Se enciende cuando el sistema está en funcionamiento
LED del sensor 1	LD-5	Se enciende a los 20 segundos (sensor de nivel bajo)
LED de velocidad 1 del motor	LD-4	Se enciende a los 22 segundos (velocidad 1 del motor)
LED del sensor 2	LD-3	Se enciende a los 45 segundos (sensor de nivel medio)
LED de velocidad 2 del motor	LD-2	Se enciende a los 47 segundos (velocidad 2 del motor)
LED del sensor 3	LD-1	Se enciende a los 70 segundos (sensor de nivel máximo)
LED de velocidad 3 del motor	LD-0	Se enciende a los 72 segundos (velocidad 3 del motor)
Displays	DISP1	Muestra el tiempo transcurrido en segundos

Tabla I: Entradas y Salidas del Sistema

II-B. Contador de tiempo.

Para llevar el control del llenado y de las velocidades de mezcla, se implementa un contador utilizando dos registros: first y second, que representan los dígitos de las unidades y decenas respectivamente. El sistema utiliza un mecanismo de retardo basado en un registro delay de 31 bits, el cual se incrementa en cada ciclo de reloj. Cuando todos los bits de delay son 1, el contador de tiempo se activa.

Cada vez que delay llega a su valor máximo, el registro first se incrementa de 0 a 9. Cuando first alcanza 9, se reinicia a 0, y el registro second se incrementa, lo que permite crear un contador de dos dígitos que va de 00 a 89. Cuando second llega a 8 y first a 9, ambos registros se reinician a 0.

II-C. Activación de sensores.

El encendido de los sensores se controla mediante el valor actual del contador. Cuando el contador alcanza ciertos valores, se activan estos Leds:

- A los 20 segundos se enciende s1 (LD-5)
- A los 45 segundos, se enciende s2 (LD-3)
- A los 70 segundos, se enciende s3 (LD-1)

Cada sensor permanece encendido indicando que se ha alcanzado ese nivel de llenado, hasta que otro sensor se activa, en ese momento se apaga el led.

II-D. Cambio de velocidades.

El cambio de velocidades del motor se refleja en los Leds:

- m1 (LD-4): Velocidad 1
- m2 (LD-2): Velocidad 2
- m3 (LD-0): Velocidad 3

El cambio de velocidades ocurre a los 2 segundos después de que se activa cada sensor:

- A los 22 segundos, se enciende m1 (LD-4), activando la velocidad 1
- A los 47 segundos, se enciende m2 (LD-2), activando la velocidad 2
- A los 72 segundos, se enciende m3 (LD-0), activando la velocidad 3

Cada cambio de velocidad apaga la anterior y enciende la nueva, así gradualmente a medida que el contenedor se llena.

De manera general, se obtiene el siguiente algoritmo.

Algorithm 1 Sistema de Llenado Automático de Leche (General)

procedure INICIALIZARSISTEMA
    Reiniciar contador de tiempo (first, second)
    Apagar todos los LEDs y sensores
end procedure
procedure ACTUALIZARCONTADOR
    if contador de retardo completo **then**
        Incrementar contador de tiempo (first, second)
        if contador llega a 89 **then**
            Reiniciar contador
        end if
    end if
end procedure
procedure CONTROLLEDSYSENSORES tiempo actual 0 segundos
    Encender LED principal 20 segundos
    Activar sensor de nivel bajo (s1) 22 segundos
    Activar velocidad 1 del motor (m1) 45 segundos
    Activar sensor de nivel medio (s2)
    Desactivar sensor de nivel bajo (s1) 47 segundos
    Activar velocidad 2 del motor (m2)
    Desactivar velocidad 1 del motor (m1) 70 segundos
    Activar sensor de nivel alto (s3)
    Desactivar sensor de nivel medio (s2) 72 segundos
    Activar velocidad 3 del motor (m3)
    Desactivar velocidad 2 del motor (m2)
    Apagar LED principal 89 segundos
    Desactivar velocidad 3 del motor (m3)
    Desactivar sensor de nivel alto (s3)
end procedure
procedure ACTUALIZARDISPLAY
    Seleccionar dígito a mostrar (first o second)
    Convertir dígito a patrón de 7 segmentos
    Activar segmentos correspondientes
end procedure
procedure PRINCIPAL
    while sistema encendido **do**
        if switch de inicio activado **then**
            ActualizarContador()
            ControlLEDsYSensores()
        else
            InicializarSistema()
        end if
        ActualizarDisplay()
    end while
end procedure
Principal()

III. ANEXO.

III-A. Código

```
`timescale 1ns / 1ps
module top(
  input clock,
  input start,      // Switch para iniciar
  output a, b, c, d, e, f, g,
  output [3:0] an,
  output led,       // LED de encendido
  output reg s1,    // Cambiado a reg
  output reg s2,    // Cambiado a reg
  output reg s3,    // Cambiado a reg
  output reg m1,    // Cambiado a reg
  output reg m2,    // Cambiado a reg
  output reg m3     // Cambiado a reg
);

reg [3:0] first;
reg [3:0] second;
reg [30:0] delay;
reg led_status;     // Estado del LED
reg s1_on, s2_on, s3_on, m1_on, m2_on, m3_on;


// Logica del contador y control de LED
always @ (posedge clock)
begin
  if (!start) begin
    delay <= 0;
    first <= 0;
    second <= 0;
    led_status <= 0;
    s1_on <= 0; s2_on <= 0; s3_on <= 0;
    m1_on <= 0; m2_on <= 0; m3_on <= 0;
  end else begin
    delay <= delay + 1;

    if (&delay) begin // Esto es equivalente a
         (delay == {23{1'b1}})
      if (first == 4'd9) begin
        first <= 0;
        if (second == 4'd8) begin
          second <= 0;
        end else begin
          second <= second + 1;
        end
      end else begin
        first <= first + 1;
      end

      // Logica de control de los LEDs
      case ({second, first})
                  8'h00: led_status <= 1;
        8'h20: s1_on <= 1;
        8'h22: m1_on <= 1;
        8'h45: begin s2_on <= 1; s1_on <= 0;
            end
        8'h47: begin m2_on <= 1; m1_on <= 0;
            end
        8'h70: begin s3_on <= 1; s2_on <= 0;
            end
        8'h72: begin
          m3_on <= 1; m2_on <= 0;
          led_status <= 0; // Apaga el LED en
              72
        end
        8'h89: begin
          m3_on <= 0; s3_on <= 0;
        end
      endcase
    end
  end
end

// Asignacion de los estados a las salidas
```

```
always @(*) begin
  s1 = s1_on;
  s2 = s2_on;
  s3 = s3_on;
  m1 = m1_on;
  m2 = m2_on;
  m3 = m3_on;
end

assign led = led_status;

// Multiplexado para la visualizacion de 7
    segmentos
localparam N = 18;
reg [N-1:0] count;
reg [6:0] sseg;
reg [3:0] an_temp;

always @ (posedge clock)
begin
  count <= count + 1;
end

always @ (*)
begin
  case (count[N-1:N-2])
    2'b00: begin
      sseg = first;
      an_temp = 4'b1110;
    end
    2'b01: begin
      sseg = second;
      an_temp = 4'b1101;
    end
    2'b10, 2'b11: begin
      sseg = 7'h3F; // Enviar "-" para los
          otros digitos
      an_temp = 4'b1011;
    end
  endcase
end

assign an = an_temp;

reg [6:0] sseg_temp;
always @ (*)
begin
  case (sseg)
    4'd0: sseg_temp = 7'b1000000; // 0
    4'd1: sseg_temp = 7'b1111001; // 1
    4'd2: sseg_temp = 7'b0100100; // 2
    4'd3: sseg_temp = 7'b0110000; // 3
    4'd4: sseg_temp = 7'b0011001; // 4
    4'd5: sseg_temp = 7'b0010010; // 5
    4'd6: sseg_temp = 7'b0000010; // 6
    4'd7: sseg_temp = 7'b1111000; // 7
    4'd8: sseg_temp = 7'b0000000; // 8
    4'd9: sseg_temp = 7'b0010000; // 9
    default: sseg_temp = 7'b0111111; // "-"
  endcase
end

assign {g, f, e, d, c, b, a} = sseg_temp;
assign dp = 1'b1; // No usar el punto decimal

endmodule
```

Asignación de pines.

```
# Boton de inicio
NET "start_button" LOC = "N3";

# Led de inicio
NET "led" LOC = "G1";

# leds
NET "s1" LOC = "N4" ;
NET "m1" LOC = "N5" ;

NET "s2" LOC = "P6" ;
NET "m2" LOC = "P7" ;

NET "s3" LOC = "M11" ;
NET "m3" LOC = "M5" ;

# Pin assignment for 7-segment displays
NET "a" LOC = "l14" ;
NET "b" LOC = "h12" ;
NET "c" LOC = "n14" ;
NET "d" LOC = "n11" ;
NET "e" LOC = "p12" ;
NET "f" LOC = "l13" ;
NET "g" LOC = "m12" ;

NET "an<3>" LOC = "k14";
NET "an<2>" LOC = "m13";
NET "an<1>" LOC = "j12";
NET "an<0>" LOC = "f12";

# Pin assignment for clock
NET "clock" LOC = "b8";
```