Trabajo Sistemas Electrónicos Digitales

Escuela Universitaria de Ingeniería Técnica Superior Industrial

Universidad Politécnica de Madrid

Desarrollo de una Cafetera en VHDL

PROGRAMACIÓN EN VHDL DE LA

NEXYS4 DDR Artix-7 FPGA

ÍNDICE

CAFETERA EN VHDL

1. OBJETIVOS

Se ha realizado la programación de una cafetera automática utilizando el lenguaje VHDL, mediante el cual se consigue el control de la cafetera para que sirva tres tipos de cafés corto, doble y largo, además de poder añadirle leche fría, templada o caliente, así como agregarle diferentes niveles de azúcar.

Utilizando la herramienta de diseño VIVADO de Xilinx inc. La cual nos permite diseñar nuestra Nexys4 DDR Artix-7 FPGA con sus respectivos módulos.

2. FUNCIONAMIENTO

La cafetera dispone de trece entradas clk, reset, encendido, corto, doble, largo, milk, leche_caliente, leche_fría, sugar, more_sugar, less_sugar y start y de trece salidas encendido_led, corto_led, doble_led, largo_led, milk_led, milk_led_red, milk_led_blue, sugar_led, bomba_led_red, done_led_green, apagado_led, display_select y display_number.

Partiendo desde la cafetera apagada con la salida apagado_led iluminada, pasamos a encenderla usando la entrada encendido, donde se activará la cafetera a través del flanco de subida del clk, iluminándose el LED correspondiente a encendido_led, mostrándonos que está encendida. De ahí pasará al estado de standby, donde esperará que se le indique el tipo de café a preparar, pudiendo ser corto, doble o largo y activando una de las entradas el LED correspondiente corto_led, doble_led o largo_led, para indicarlo. Además, podremos ver en los dos displays de más a la derecha el tiempo que se tarda en preparar cada café, a saber, 10 segundos para corto, 15 segundo para doble y 20 segundos para largo. Si activamos dos tipos o más de café a la vez, saldrá una cuenta de 0 segundos en la pantalla y el botón de start estará bloqueado.

En este mismo estado de standby también existe la opción de poder elegir si queremos leche o azúcar. Si elegimos leche, se activará el LED milk_led, en el display podremos ver la letra L y un guión, usando los botones de leche_cliente o leche_fría podemos elegir su temperatura, si no se pulsa ninguno, la leche saldrá del tiempo. Si elegimos fría se encenderá la salida milk_led_blue y el display con el guión pasará a ser una F y si es caliente la opción de milk_led_red y el display con el guión pasará a ser una C. Como opción especial, sin seleccionar ningún tipo de café, pero activando la opción de leche y dando a start, podremos elegir un vaso de leche que tardará 5 segundos en servirse. Por otro lado, si activamos la entrada sugar, se activará el LED de salida sugar_led y se activarán los 4 displays de más a la izquierda, donde podremos elegir el nivel de azúcar usando los botones de more_sugar o less_sugar. Una vez tengamos el café a nuestro gusto, presionando el botón de start comenzará la cuenta atrás.

Inmediatamente después de presionar start pasará al estado sirviendo, activándose el LED de salida bomba_led_rojo, el cual parpaderá durante la cuenta atrás para servirse el café. La cuenta regresiva la podremos ver en los displays de la tarjeta Nexys4, activados por las salidas display_select y display_number. Una vez terminado el café se encenderá la salida done_led_green y se volverá al estado de standby, quedando encendidas las opciones que no hayan sido desactivadas y esperando para servir un nuevo café.

Durante la cuenta atrás si se pulsa cualquier entrada, ya sea corto, doble, largo o las opciones de leche o azucar, la cafetera no hará nada hasta que finalice el tiempo de servido. Si en cualquier momento se activa el botón reset asíncrono o se apaga la entrada de encendido, se activará el LED de salida apagado_led y se apagarán todos los displays y los LEDs, indicando que la cafetera está apagada. El sistema retornará al estado de encendida una vez la señal de encendido esté activa y la señal de reset apagada.

3. MÓDULOS

Mediante los siguientes diagramas podemos ver la interconexión de los diferentes módulos y como se distribuyen dentro de la cafetera para ejecutar su correcto funcionamiento. La cafetera consta de los siguientes módulos:

- Cafetera
- Divisor de frecuencia.
- Antirebote.
- Máquina de estados.
- Decodificador.
- Refresco de displays.

Cada módulo tiene su respectivo testbench para comprobar la funcionalidad del módulo, así como un testbench para la cafetera que los testea a todos a la vez. A continuación se explicará cada módulo, con su correspondiente diagrama, código y simulación del testbench.

3.1. Cafetera

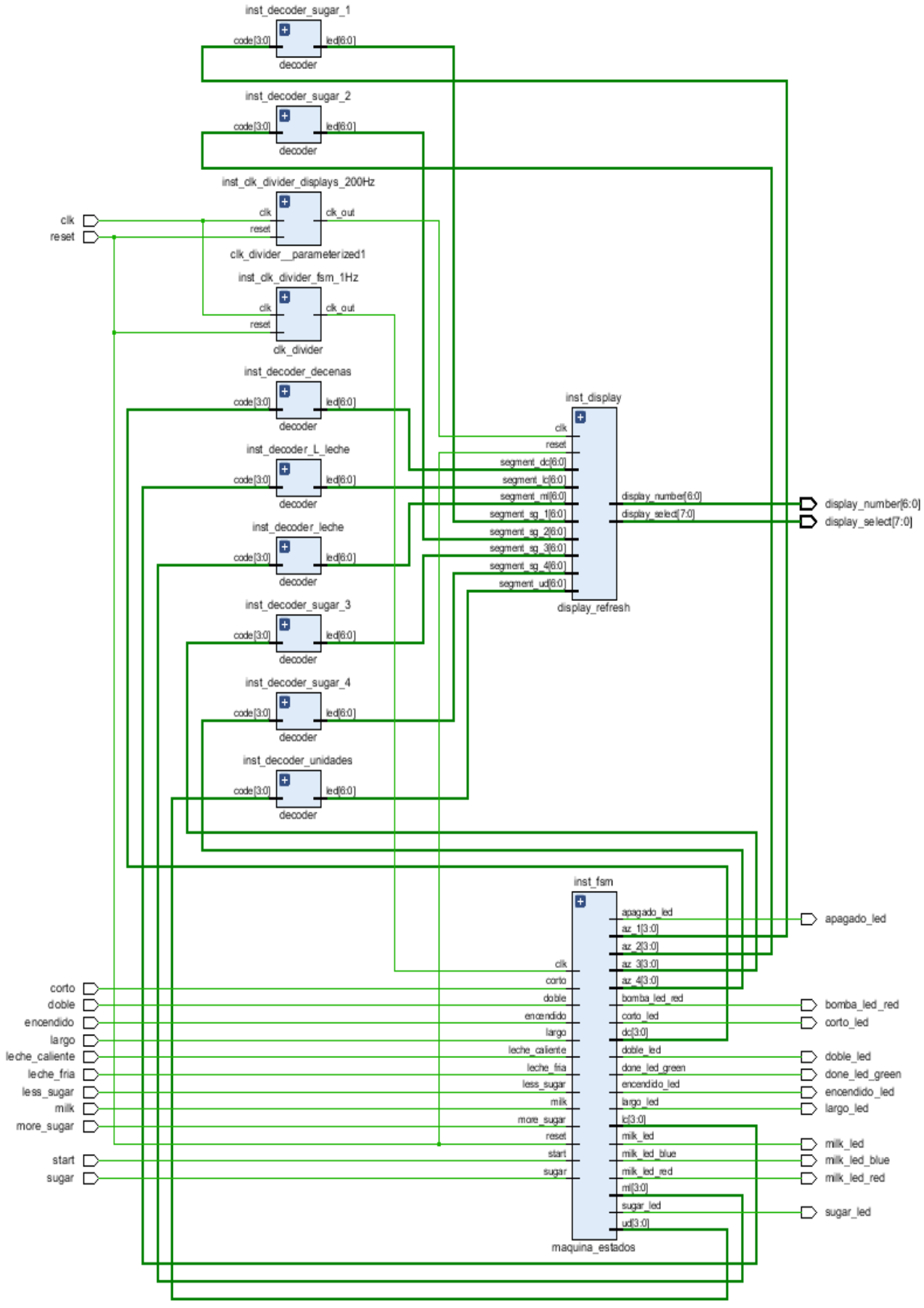
3.1.1. Diagrama General de la Cafetera

Este primer módulo es el encargado de instanciar todos los demás módulos del programa, llamado módulo de cabecera, el cual sincroniza todas las entradas y salidas de cada una de las demás instanciaciones de los módulos, para el correcto desarrollo del sistema. Todas las entrada y salidas que interactúan con el usuario están definidas en él.

En el siguiente diagrama podemos ver la interrelación de los módulos y sus instanciamientos, que constituyen nuestra cafetera. Disponemos de un módulo divisores de frecuencia instanciado dos veces, los cuales proporcionan diferentes frecuencias a los módulos máquina de estados y refresco de displays, respectivamente. Hay un módulo debounce instanciado cinco veces para evitar el rebote mecánico de los botones y posibles falsas señales. También dispones de un módulo decodificador instanciado ocho veces, uno por cada display de la placa. Un módulo máquina de estados, encargado de la gestión de los diferentes estados de la cafetera. Por último, al final del diagrama tenemos un módulo de refresco de displays, encargado de iluminar los displays alternadamente e iluminarlos.

Las señales de entrada van desde el usuario al módulo cafetera, el cual las distribuye internamente a los demás módulos y estos tras procesarlas dan una señal de salida que devuelven al módulo cafetera y de ahí al usuario.

El orden lógico de los módulos dentro de la cafetera es el siguiente: el divisor de frecuencia aporta relojes específicos a la máquina de estados y al refresco de displays, los botones de entrada pasan por el debounce y van a la máquina de estados. De la máquina de estados se leen las salidas de los LEDs hacia las salidas del módulo cafetera y se pasan los códigos para los displays al decodificador. Finalmente, del decodificador se pasa al refresco de displays, el cual enciende los segmentos adecuados de cada display en función del código proveniente del decodificador. El refresco de displays devuelve la señal de salida al módulo de cafetera para que le llegue al usuario.


inst_decoder_sugar_1
code[3:0]
led[6:0]
decoder
inst_decoder_sugar_2
code[3:0]
led[6:0]
decoder
inst_clk_divider_displays_200Hz
clk
reset
clk_out
clk_divider__parameterized1
clk
reset
inst_clk_divider_fsm_1Hz
clk
reset
clk_out
clk_divider
inst_decoder_decenas
code[3:0]
led[6:0]
decoder
inst_decoder_L_leche
code[3:0]
led[6:0]
decoder
inst_decoder_leche
code[3:0]
led[6:0]
decoder
inst_decoder_sugar_3
code[3:0]
led[6:0]
decoder
inst_decoder_sugar_4
code[3:0]
led[6:0]
decoder
inst_decoder_unidades
code[3:0]
led[6:0]
decoder
inst_display
clk
reset
segment_dc[6:0]
segment_lc[6:0]
segment_mi[6:0]
segment_sg_1[6:0]
segment_sg_2[6:0]
segment_sg_3[6:0]
segment_sg_4[6:0]
segment_ud[6:0]
display_number[6:0]
display_select[7:0]
display_refresh
display_number[6:0]
display_select[7:0]
inst_fsm
clk
corto
doble
encendido
largo
leche_caliente
leche_fria
less_sugar
milk
more_sugar
reset
start
sugar
apagado_led
az_1[3:0]
az_2[3:0]
az_3[3:0]
az_4[3:0]
bomba_led_red
corto_led
dc[3:0]
doble_led
done_led_green
encendido_led
largo_led
lc[3:0]
milk_led
milk_led_blue
milk_led_red
mi[3:0]
sugar_led
ud[3:0]
maquina_estados
corto
doble
encendido
largo
leche_caliente
leche_fria
less_sugar
milk
more_sugar
start
sugar
apagado_led
bomba_led_red
corto_led
doble_led
done_led_green
encendido_led
largo_led
milk_led
milk_led_blue
milk_led_red
sugar_led

3.1.2. Código Cafetera

En la entidad declaramos las variables físicas del sistema mediante entradas y salidas de la cafetera, anteriormente descritas.

```
entity Cafetera is
    PORT ( clk : in STD_LOGIC; -- 100 MHz
           reset : in STD_LOGIC;
           encendido : in STD_LOGIC;
           corto : in STD_LOGIC;
           doble : in STD_LOGIC;
           largo : in STD_LOGIC;
           milk : in STD_LOGIC;
           leche_caliente : in STD_LOGIC;
           leche_fria : in STD_LOGIC;
           sugar : in STD_LOGIC;
           more_sugar: in STD_LOGIC;
           less_sugar : in STD_LOGIC;
           start : in STD_LOGIC;
           encendido_led : out STD_LOGIC;
           corto_led : out STD_LOGIC;
           doble_led : out STD_LOGIC;
           largo_led : out STD_LOGIC;
           milk_led : out STD_LOGIC;
           milk_led_red : out STD_LOGIC;
           milk_led_blue : out STD_LOGIC;
           sugar_led : out STD_LOGIC;
           bomba_led_red : out STD_LOGIC;
           done_led_green : out STD_LOGIC;
           apagado_led : out STD_LOGIC;
           display_select : out STD_LOGIC_VECTOR (7 downto 0);
           display_number : out STD_LOGIC_VECTOR (6 downto 0)
           );
end Cafetera;
```

En la arquitectura instanciamos los diferentes componentes que compondrán la cafetera, es decir, los módulos, con sus correspondientes entradas y salidas, aquí se determinarán el número de los mismo en el proyecto. Así también, se generan los mapas de los puertos de entradas y salidas entre módulos para conectarlos entre ellos y enviar las salidas de unos a las entradas de otros, y permitir el correcto fluir de los valores a través del código.

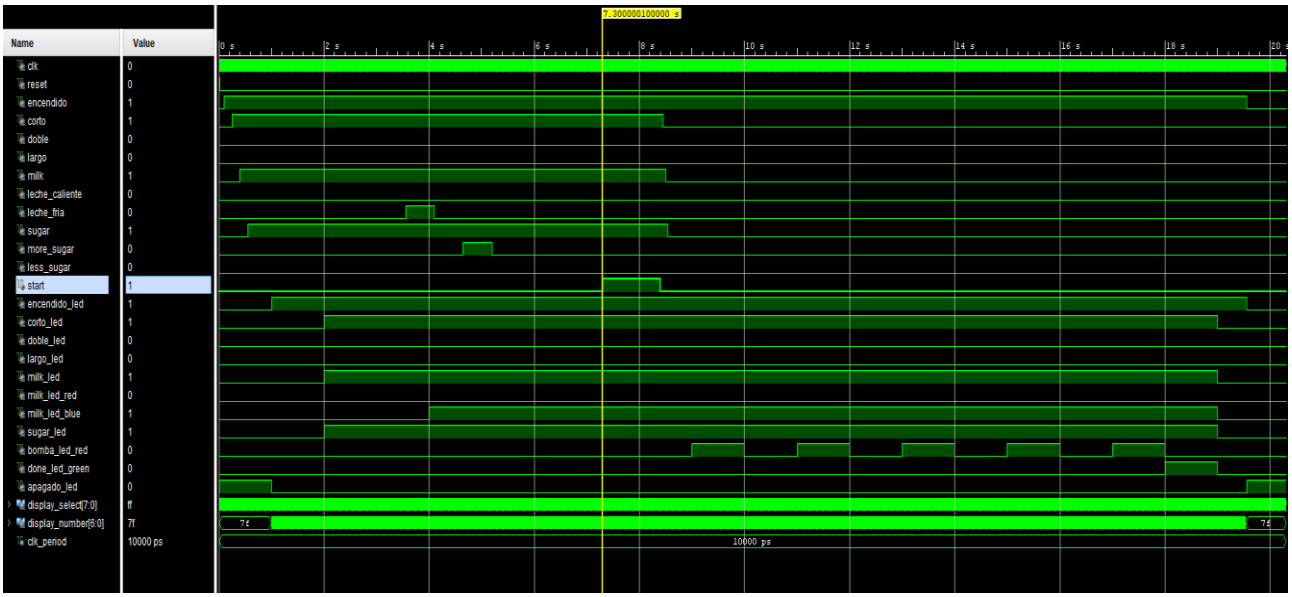
Cabe destacar los dos módulos divisores de frecuencia, a los cuales les inicializamos el valor de relación a través del mapa genérico, para conseguir frecuencias distintas en sus respectivas salidas, a saber, 1 Hz o 400 Hz. La salida de 1 Hz irá al módulo instanciado de la máquina de estados y la salida de 400 Hz irá al módulo del display_refresh.

```
    -- INSTANCIACION COMPONENTES

BEGIN

    inst_clk_divider_fsm_1Hz: clk_divider
        GENERIC MAP( relacion => 50000000) -- 1 Hz a fsm
        PORT MAP(
            -- in
            clk => clk, -- 100 MHz
            reset => reset,
            -- out
            clk_out => clk_fsm -- 1 Hz
            );

    inst_clk_divider_displays_400Hz: clk_divider
        GENERIC MAP( relacion => 125000) -- 400 Hz a displays
        PORT MAP(
            -- in
            clk => clk, -- 100 MHz
            reset => reset,
            -- out
            clk_out => clk_display -- 400 Hz
            );
```

También instanciaremos ocho módulos decodificadores, uno para cada display (unidades, decenas, leche y azúcar) y cinco instanciaciones llamadas debounce (antirebote), el módulo máquina de estados y el de refresco de displays.

3.1.3. Simulación Cafetera

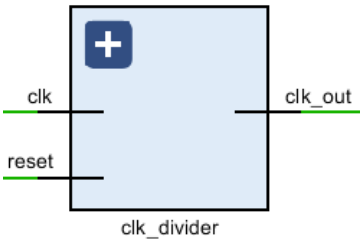
En la siguiente simulación podemos ver las diferentes entradas y salidas durante la preparación de un café corto con leche fría y dos niveles de azúcar. Primero se activado encendido para encender la cafetera. Seguidamente se activa la entrada corto, milk y sugar. Por último, se simula un impulso de leche_fría y tres de more_sugar para el nivel dos de azúcar.



3.2. Divisor de Frecuencia

3.2.1. Diagrama Divisor de Frecuencia

Este módulo se encarga de dividir la frecuencia que recibe de la placa Nexys4 (100MHz) a unas frecuencias mucho más reducidas para el correcto funcionamiento de los módulos máquina de estados (1Hz) y refresco de displays (400Hz).



3.2.2. Código Divisor de Frecuencia

Como se puede ver en el diagrama, se dispone de dos entradas, una para el clk y otra para el reset. Respecto a la salida, solo dispone de clk_out, por la cual saldrá la frecuencia divida al valor que necesitemos.

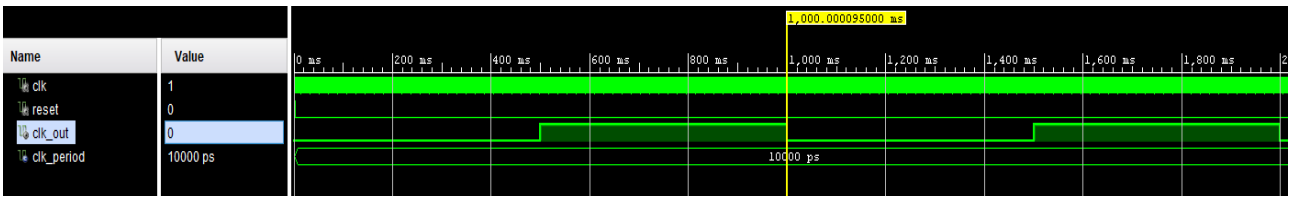
```
entity clk_divider is
    generic ( relacion: integer := 10000000);
    PORT ( clk : in  STD_LOGIC; -- 100 MHz
           reset : in  STD_LOGIC;
           clk_out : out  STD_LOGIC
           );
end clk_divider;

architecture Behavioral of clk_divider is

    signal count: integer range 1 to relacion;
    signal clk_out_i: STD_LOGIC := '0';

BEGIN

    frequency_divider: process (clk , reset)
    BEGIN
       if reset = '1' then
           count <= 1;
           clk_out_i <= '0';
       elsif clk = '1' and clk'event then
           if (count = relacion) then
               count <= 1;
               clk_out_i <= not (clk_out_i);
           else
               count <= count + 1;
           end if;
        end if;
    end process;

    clk_out <= clk_out_i ;

end Behavioral;
```

Usando la señal count en un rango de 0 a relación, hacemos una cuenta de los ciclos del reloj hasta el valor que de relación, pasado a través del módulo de cafetera, depende de la instanciación utilizada, será un valor diferente. En ese momento nuestra señal clk_out_i cambia su estado entre 0 y 1 simulando a un reloj de menor frecuencia. Esta señal clk_out_i es la que asignaremos a la salida del módulo clk_out. Para conseguir la frecuencia de 1 Hz, relación valdrá 50000000, es decir, medio segundo y para 400 Hz relación valdrá 125000, es decir, 1/400 segundos.

3.2.3. Simulación Divisor de frecuencia

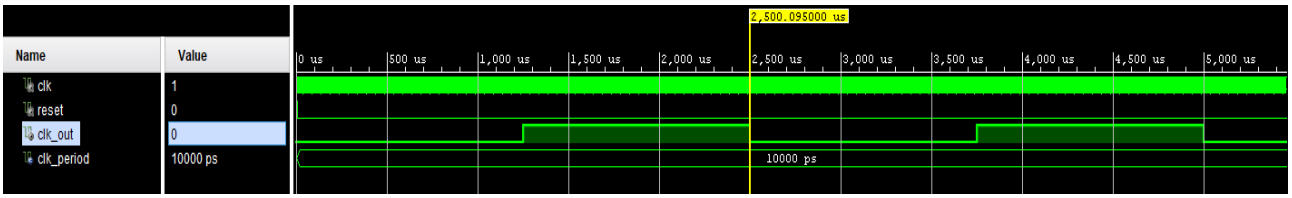
3.2.3.1. Frecuencia 1 Hz

En esta primera simulación se estable la variable relación desde el mapa genérico entre la cafetera y el divisor de frecuencia a 50000000 para simular un semiciclo de 500 ms con salida baja y otro semiciclo de 500 ms a salida alta, para dividir el reloj de 100 MHz a 1Hz.



3.2.3.2. Frecuencia 400 Hz

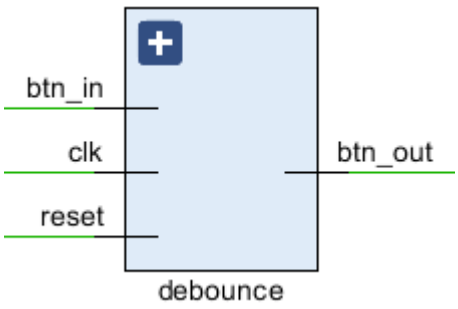
En esta segunda simulación se hace lo mismo que en la anterior pero el valor de relación es de 125000, dividendo los 100 MHz de entrada en dos semiciclos de 1250 microsegundos, para tener una salida de 400 HZ.



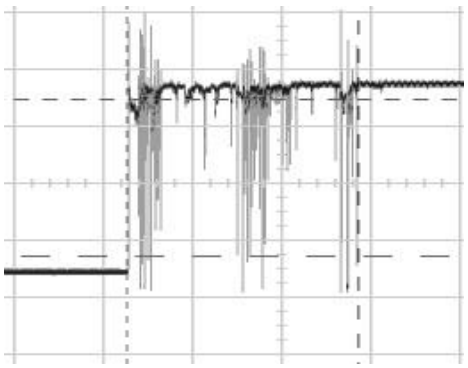
3.3. Debounce

3.3.1. Diagrama Debounce

Este módulo se encarga del antirebote de los botones físicos de la placa, evitando así que al mantenerlos se interprete una pulsación múltiple. Instanciaremos este módulo cinco veces, para los botones more_sugar, less_sugar, leche_caliente, leche_fría y start.



Los pulsadores mecánicos son baratos, pero al ser accionados tardan unos milisegundos en estabilizarse, sufren de bouncing o rebote, pudiendo cambiar de estado durante estos segundos y aportando malas lecturas. Aunque el tiempo es muy bajo, el sistema es capaz de reconocerlo. La siguiente imagen muestra una señal de rebote típica.



Nuestra solución a nivel de software se encarga de recordar si el botón estaba pulsado o no en un momento anterior y comprobar el estado actual, como en los biestables o flip flops. La idea general es que si durante un periodo de tiempo lo suficientemente amplio, unos 15 milisegundos en nuestro caso, el valor actual del pulsador y el valor que almacena el estado anterior son iguales, podemos dar por sentado que el pulsador ha llegado a un estado de pulsación estable.

3.3.2. Código Debounce

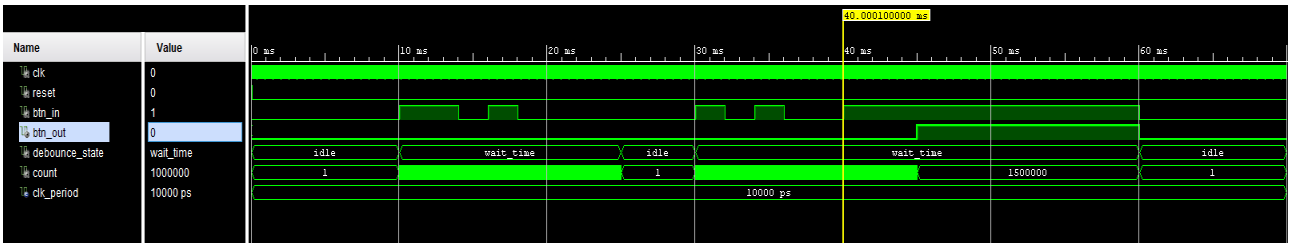
El siguiente código VHDL implementa el circuito antes explicado:

```
BEGIN

    -- RESET Y PASO DE ESTADO
    debounce_btn: process(reset,clk)
    BEGIN

        if reset = '1' then
            debounce_state <= idle;
            btn_out <= '0';

        -- Asginacion estados
        elsif clk = '1' and clk'event then
            case debounce_state is

                WHEN idle =>
                    if btn_in = BTN_ACTIVE then
                        debounce_state <= wait_time;
                    else
                        debounce_state <= idle; --wait until button is pressed.
                    end if;
                    btn_out <= '0';

                WHEN wait_time =>
                    if count = COUNT_MAX then
                        if btn_in = BTN_ACTIVE then
                            btn_out <= '1';
                            debounce_state <= wait_time;
                        else
                            count <= 1;
                            debounce_state <= idle;
                            --btn_out <= '0';
                        end if;
                    else
                        count <= count + 1;
                        --btn_out <= '0';
                    end if;

                WHEN OTHERS => NULL;

            end case;
        end if;
    end process;
```

Como se puede ver en el código se dispone de dos estados, uno de reposo y el otro para hacer la cuenta. Más concretamente, en el primer estado idle, se espera que el botón sea presionado para cambiar al estado wait_time. En este segundo estado se inicia una cuenta a través del contador count, que ascenderá hasta alcanzar el valor de COUNT_MAX, el cual vale 1500000, que teniendo en cuenta el reloj del módulo, equivalen a 15 ms, suficiente para evitar el rebote de los botones. En caso de alcanzar ese valor, se módulo tendrá como salida un valor de 1, hasta que se deje de pulsar el botón, en dicho momento la salida será de 0 de nuevo. Se ha hecho de este modo para poder mantener el botón de azúcar y el valor vaya cambiando de forma ascendente o descendente sin la necesidad de pulsar repetidas veces.

3.3.3. Simulación Debounce

En esta simulación podemos ver varios pulsos en un primer intento de activar el botón de entrada durante menos de 10 ms, no obteniendo un valor de salida. En un segundo intento hay dos pulsos cortos y un tercer pulso más largo que se mantiene, entre el primer pulso y el última de este segundo intento hay más de 10 ms, por lo que la salida si es activada. También se puede ver el cambio de estado entre idle y wait_time.



3.4. Máquina de estados

3.4.1. Diagrama Máquina de Estados

Este módulo es el encargado de gestionar el estado en el que se encuentra la cafetera en todo momento, variando entre apagado, standby y sirviendo. La cafetera estará en el estado apagado, cuando reset valga cero y encendido uno, se parará al estado standby. En el estado standby queda a la espera de un de las entradas de leche y/o café. Por último, está el estado sirviendo, en el cual la cafetera estará haciendo la cuenta atrás para servir el tipo de café. Una vez finalizado volverá al estado de standby.

3.4.2. Código Máquina de Estados

En esta primera parte del código nos encontramos con la entidad, con sus entradas y salidas correspondientes. Las entradas provienen del módulo cafetera y del divisor de frecuencia, está última le proporcionará 1 Hz a la entrada clk. Respecto a las salidas se tratan de los LEDs que nos irán indicando las selecciones que hagamos en la cafetera en todo momento. Hay ocho de las salidas que irán a los decodificadores. La salida ud manda al decodificador el valor de las unidades, la salida dc corresponde a las decenas, la salida ml corresponde a la temperatura de la leche, la salida lc corresponde a la letra L y las salidas az_1, az_2, az_3 y az_4 corresponden a las salidas de los niveles de azúcar.

```
entity maquina_estados is
    PORT ( clk : in STD_LOGIC; -- 1 Hz
           reset : in STD_LOGIC;
           encendido : in STD_LOGIC;
           corto : in STD_LOGIC;
           doble : in STD_LOGIC;
           largo : in STD_LOGIC;
           milk : in STD_LOGIC;
           leche_caliente : in STD_LOGIC;
           leche_fria : in STD_LOGIC;
           sugar : in STD_LOGIC;
           more_sugar : in STD_LOGIC;
           less_sugar : in STD_LOGIC;
           start : in STD_LOGIC;
           encendido_led : out STD_LOGIC;
           corto_led : out STD_LOGIC;
           doble_led : out STD_LOGIC;
           largo_led : out STD_LOGIC;
           milk_led : out STD_LOGIC;
           milk_led_red : out STD_LOGIC;
           milk_led_blue : out STD_LOGIC;
           sugar_led : out STD_LOGIC;
           bomba_led_red : out STD_LOGIC;
           done_led_green : out STD_LOGIC;
           apagado_led : out STD_LOGIC;
           ud : out STD_LOGIC_VECTOR (3 downto 0);
           dc : out STD_LOGIC_VECTOR (3 downto 0);
           ml : out STD_LOGIC_VECTOR (3 downto 0);
           lc : out STD_LOGIC_VECTOR (3 downto 0);
           az_1 : out STD_LOGIC_VECTOR (3 downto 0);
           az_2 : out STD_LOGIC_VECTOR (3 downto 0);
           az_3 : out STD_LOGIC_VECTOR (3 downto 0);
           az_4 : out STD_LOGIC_VECTOR (3 downto 0)
           );
end maquina_estados;
```

En la arquitectura de este módulo nos encontramos con las señales, estas las utilizaremos en el código de este módulo para poder crear las condiciones que requiramos para su correcto funcionamiento y finalmente asignarlas a las señales de salida del módulo. En los siguientes extractos del código iremos viendo como desarrollado las diferentes funcionalidades de la cafetera utilizando estas señales.

En este primer proceso nos encargaremos de pasar al estado de apagado en el caso de recibir la entrada reset activada o encendido desactivada. En el caso de evolución normal de la cafetera, cambiaremos de estado según corresponda en cada ciclo de reloj. Este proceso es el encargado de ejecutar las diferentes acciones en función del estado actual e ir cambiando entre estos según las entradas que recibamos y el valor de las señales en dicho momento.

Utilizando este tipo de estructura en el código, nos ahorramos el tener que declarar el estado o valor de cada variable en cada proceso, pudiendo guardar dicho estado o valor entre ciclos del reloj.

```
-- RESET Y PASO DE ESTADO
state_register :process (reset,clk,encendido)
BEGIN

    if reset = '1' or encendido = '0' then
        current_state <= inicio;
        corto_led_i <= '0';
        doble_led_i <= '0';
        largo_led_i <= '0';
        coffee_time <= init_cafe;
        milk_led_i <= '0';
        current_state_milk <= init_milk;
        temperatura_leche <= 0;
        milk_led_red_i <= '0';
        milk_led_blue_i <= '0';
        letra_L_leche <= '0';
        sugar_led_i <= '0';
        current_state_sugar <= init_sugar;
        level_sugar <= 0;
        bomba_led_red_i <= '0';
        done_led_green_i <= '0';

    elsif clk = '0' and clk'event then
        current_state <= next_state;
        corto_led_i <= corto_led_i_next;
        doble_led_i <= doble_led_i_next;
        largo_led_i <= largo_led_i_next;
        coffee_time <= coffee_time_next;
        milk_led_i <= milk_led_i_next;
        current_state_milk <= next_state_milk;
        temperatura_leche <= temperatura_leche_next;
        milk_led_red_i <= milk_led_red_i_next;
        milk_led_blue_i <= milk_led_blue_i_next;
        letra_L_leche <= letra_L_leche_next;
        sugar_led_i <= sugar_led_i_next;
        current_state_sugar <= next_state_sugar;
        level_sugar <= level_sugar_next;
        bomba_led_red_i <= bomba_led_red_i_next;
        done_led_green_i <= done_led_green_i_next;
    end if;
end process;
```

En este segundo proceso se decidirá el estado principal de la cafetera en todo momento. En el estado inicio el tiempo se le pasará el tipo select_cafe y se pasará al estado standby. En este esado se esperará la entrada de un tipo de café o la entrada única de leche. Una vez seleccionado una de estas entradas, se pasará al estado sirviendo cuando se presione el botón start. En el estado sirviendo se esperará a que se active la señal de ok para volver al estado standby.

```
BEGIN

    -- Para quitar latch
    next_state <= current_state;
    coffee_time_next <= coffee_time;

    case current_state is

        WHEN inicio =>
            if reset = '0' and encendido = '1' then
                coffee_time_next <= select_cafe;
                next_state <= standby;
            end if;

        WHEN standby =>
            if start = '0' then
                if corto = '1' and doble = '0' and largo = '0' then
                    coffee_time_next <= cafe_corto;
                elsif corto = '0' and doble = '1' and largo = '0' then
                    coffee_time_next <= cafe_doble;
                elsif corto = '0' and doble = '0' and largo = '1' then
                    coffee_time_next <= cafe_largo;
                else
                    if corto = '0' and doble = '0' and largo = '0' and milk = '1' then
                        coffee_time_next <= only_milk;
                    else
                        coffee_time_next <= select_cafe;
                    end if;
                end if;
            elsif start = '1' then
                if coffee_time_next /= select_cafe or coffee_time_next = only_milk then
                    next_state <= sirviendo;
                end if;
            end if;

        WHEN sirviendo =>
            if ok = '1' then -- senal de la cuenta atras
                coffee_time_next <= select_cafe;
                if start = '0' then
                    next_state <= standby;
                end if;
            end if;

        WHEN OTHERS => NULL;

    end case;
end process;
```

Ahora nos encargaremos de asignar el valor a las señales que más adelante asignaremos a sus respectivas salidas, siendo directamente asignadas estas salidas los LEDs de nuestra placa Nexys4 DDR justo después de este proceso. Dichas asignaciones dependerán del estado en el que se encuentre en ese momento el sistema.

Para la asignación de los LEDs de milk, se han creado dos estados, init_milk en el que se inician todos los valores de las salidas y un segundo slect_milk para seleccionar la temperatura de la leche mediante las entradas. Si la entrada milk esta apagada, entonces todas las salidas son igual a cero.

```
-- Milk LED
if milk = '1' then
    milk_led_i_next <= '1';
    letra_L_leche_next <= '1';
    -- Estados Leche
    case current_state_milk is

        WHEN init_milk =>
            temperatura_leche_next <= 3;
            milk_led_red_i_next <= '0';
            milk_led_blue_i_next <= '0';
            next_state_milk <= select_milk;

        WHEN select_milk =>
            -- Temperatura Leche
            if leche_caliente = '1' and leche_fria = '0' then
                temperatura_leche_next <= 1;
                milk_led_red_i_next <= '1';
                milk_led_blue_i_next <= '0';
            elsif leche_fria = '1' and leche_caliente = '0' then
                temperatura_leche_next <= 2;
                milk_led_red_i_next <= '0';
                milk_led_blue_i_next <= '1';
            end if;

        WHEN OTHERS => NULL;

    end case;
else
    milk_led_i_next <= '0';
    letra_L_leche_next <= '0';
    temperatura_leche_next <= 0;
    milk_led_red_i_next <= '0';
    milk_led_blue_i_next <= '0';
    next_state_milk <= init_milk;
end if;
```

Para la asignación de los LEDs de sugar, se han hecho lo mismo que para los LEDs de milk pero en este caso en el estado de slect_sugar se cambia el valor de level_sugar en función de la entrada de more_sugar o less_sugar introducido por el usuario para variarlo.

```
-- Sugar LED
if sugar = '1' then
    sugar_led_i_next <= '1';
    -- Estados Azucar
    case current_state_sugar is

        WHEN init_sugar =>
            level_sugar_next <= 1;
            next_state_sugar <= select_sugar;

        WHEN select_sugar =>
            -- Nivel Azucar
            if more_sugar = '1' and less_sugar = '0' then
                if level_sugar = 1 then
                    level_sugar_next <= 2;
                elsif level_sugar = 2 then
                    level_sugar_next <= 3;
                elsif level_sugar = 3 then
                    level_sugar_next <= 4;
                end if;
            elsif more_sugar = '0' and less_sugar = '1' then
                if level_sugar = 4 then
                    level_sugar_next <= 3;
                elsif level_sugar = 3 then
                    level_sugar_next <= 2;
                elsif level_sugar = 2 then
                    level_sugar_next <= 1;
                end if;
            end if;

        WHEN OTHERS => NULL;

    end case;
else
    sugar_led_i_next <= '0';
    level_sugar_next <= 0;
    next_state_sugar <= init_sugar;
end if;
```

Para los LEDs del tipo de café se han creado estas sentencias if. Haciéndolo de este modo, podremos ver en el mismo momento que activamos las entradas de tipo de café un LED de encendido sobre dicha entrada, aprovechando que todos los procesos se ejecutan concurrentemente y dándonos la posibilidad de ver estos valores en tiempo real. También se aplica esta idea a los displays y los LEDs de temperatura de la leche, para que el usuario pueda ver una respuesta a sus acciones ante las entradas seleccionadas en cada momento.

```
-- Cafe corto LED
if corto = '1' then
    corto_led_i_next <= '1';
else
    corto_led_i_next <= '0';
end if;

-- Cafe doble LED
if doble = '1' then
    doble_led_i_next <= '1';
else
    doble_led_i_next <= '0';
end if;

-- Cafe largo LED
if largo = '1' then
    largo_led_i_next <= '1';
else
    largo_led_i_next <= '0';
end if;
```

En cuanto al LED de bomba_led_red durante el estado sirviendo, por cada segundo que pasa en la cuenta atrás el led cambiará entre encendido y apagado, dando el efecto de parpadear durante mientras se sirve el café.

```
-- Blinking bomba LED
case bomba_led_red_i is

    WHEN '0' =>
        bomba_led_red_i_next <= '1';

    WHEN '1' =>
        bomba_led_red_i_next <= '0';

    WHEN OTHERS => NULL;

end case;
```

En el siguiente proceso se asigna el tiempo a un contador, encargado de hacer la cuenta regresiva del tiempo que se tarda en preparar el café utilizando la señal count en cada ciclo del reloj. El valor del contador dependerá del estado guardado en la variable coffe_time, asignada en el primer proceso de Paso de Estado. Cuando el café está listo, activaremos la señal de ok, utilizada en el primer proceso para volver al estado de standby.

```
-- CONTADOR Y ASIGNACION OK
contador: process (clk,current_state,next_state,coffee_time,count,ok)
BEGIN

    if current_state = standby then
        ok <= '0';
        case coffee_time is
            WHEN init_cafe => count <= 0;
            WHEN select_cafe => count <= 0;
            WHEN cafe_corto => count <= 10;
            WHEN cafe_doble => count <= 15;
            WHEN cafe_largo => count <= 20;
            WHEN only_milk => count <= 5;
            WHEN OTHERS => NULL;
        end case;

    elsif current_state = sirviendo then
        if clk = '0' and clk'event then
            count <= count - 1;
            ok <= '0';
            if count = 2 then -- para sincronizar delays por cambios de estado
                ok <= '1';
            elsif count <= 0 then
                count <= 0;
            end if;
        end if;
    end if;
end process;
```

Para mostrar el tiempo en los displays necesitaremos separar las unidades de las decenas, para poder asignar dichas señales a las salidas de esté modulo y enviarlas a los decodificadores. El valor de las unidades lo conseguimos utilizando el operador rem 10 (remainder) devuelve el resto de count cuando se divide entre 10. Obtenemos las decenas dividiendo por 10 el valor del contador count. En caso de estar en el estado inicio se carga el número 10, el cual el decodificador lo interpreta como todos los segmentos apagados.

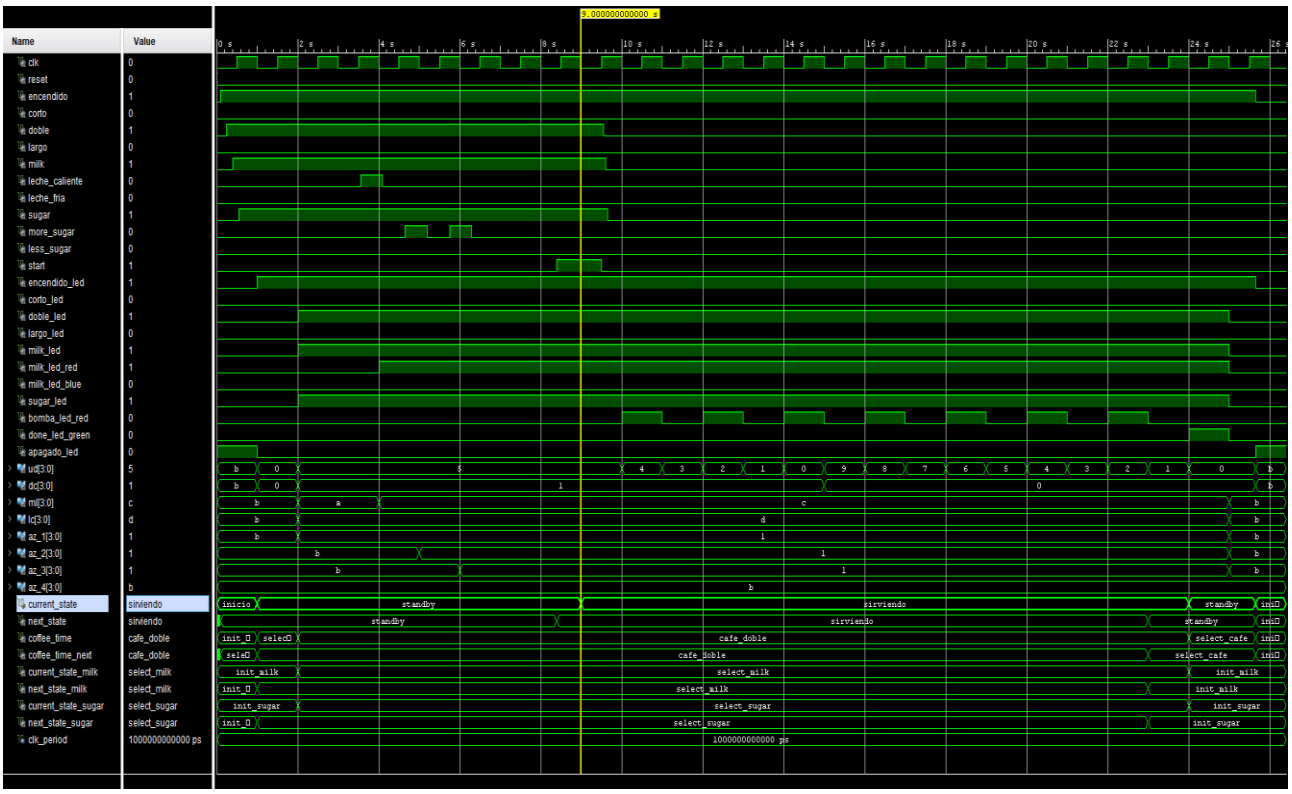
```
-- UNIDADES Y DECENAS
asigancion_unidades_decena : process (current_state,count,decena)
BEGIN

    if current_state = standby or current_state = sirviendo then
        unidad <= count rem 10;
        decena <= count / 10;
    else
        unidad <= 10;
        decena <= 10;
    end if;
end process;
```

Finalmente cambiamos de base las variables que se van a mandar a los decodificadores desde decimales enteros a binario, para que el decodificador pueda entender los valores de estas señales, una vez se las mandemos a través de las salidas correspondientes, anteriormente nombradas.

3.4.3. Simulación Máquina de Estados

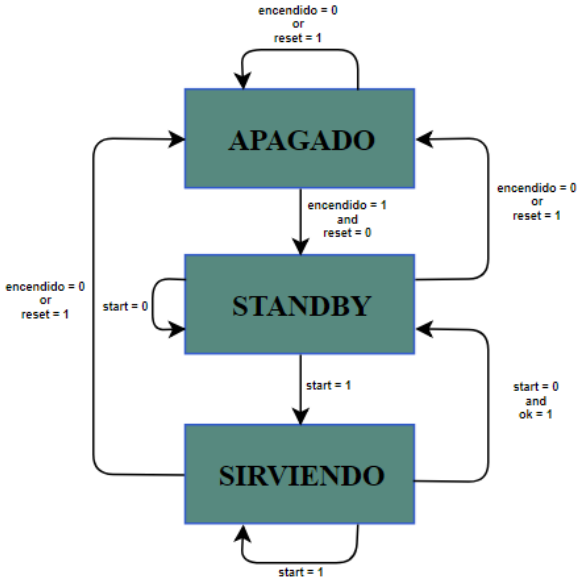
Durante esta simulación se prepara un café doble con leche caliente y tres niveles de azúcar. Primero se activa la entrada encendido para activar la cafetera. Después se activa la entrada doble, activándose el LED doble_led y saliendo 15 segundos en los displays, también se activan las entradas milk y sugar, viéndose como se ponen a uno los LEDs milk_led y sugar_led. Seguidamente se pone a uno la entrada leche_caliente durante 550 ms, se ve como se activa el LED rojo milk_led_red. Continuamos con tres pulsos de 550 ms de la entrada more_sugar para poner el nivel tres de azúcar.

Se puede ver en la parte inferior como varían los valores de los displays en función de las entradas y la cuenta atrás. En la parte más baja podemos ver como varían la cafetera entre sus estados, además de los estados del tiempo que cargamos en los displays, y la variación del estado del azúcar y de la leche entre su estado de inicio y su estado de selección.

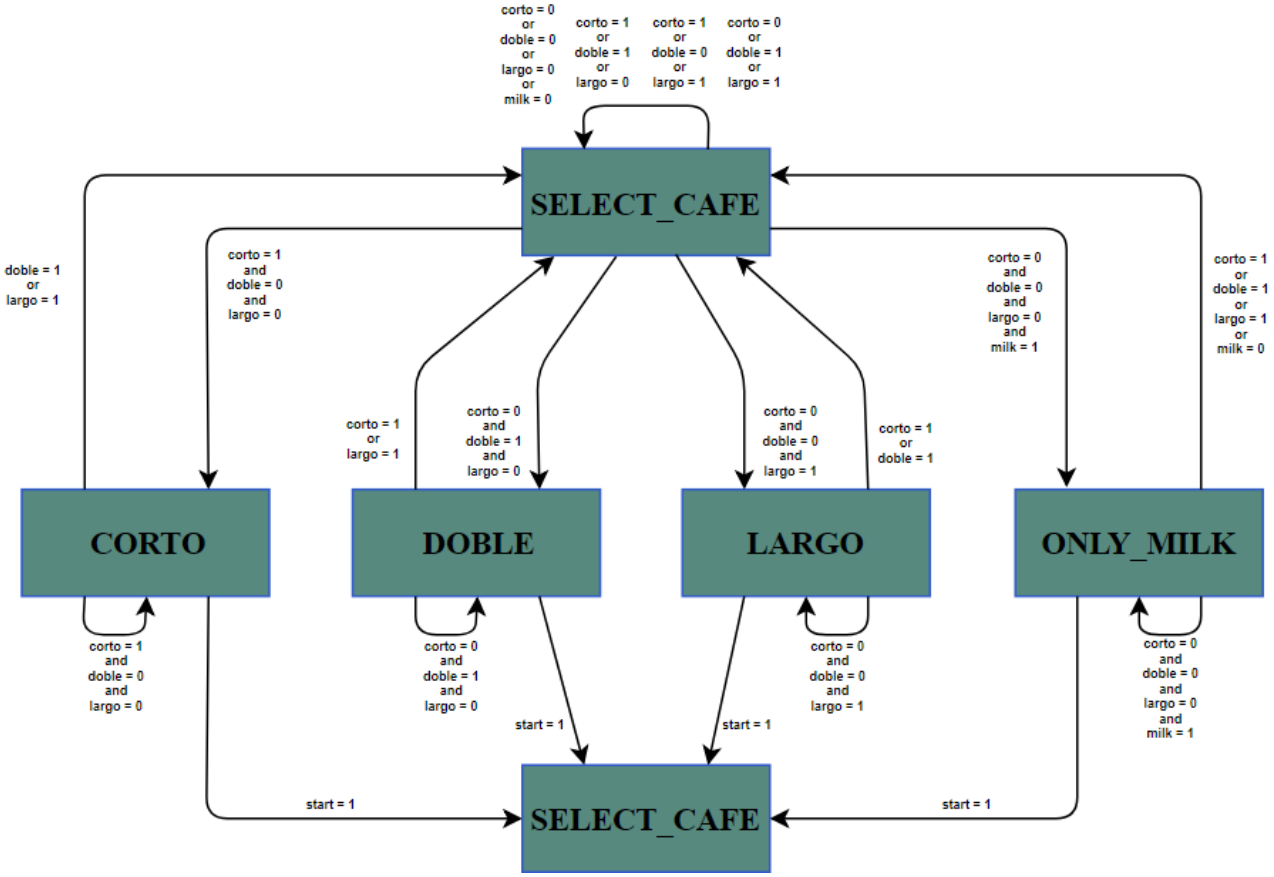


3.4.4. Diagrama de Estados

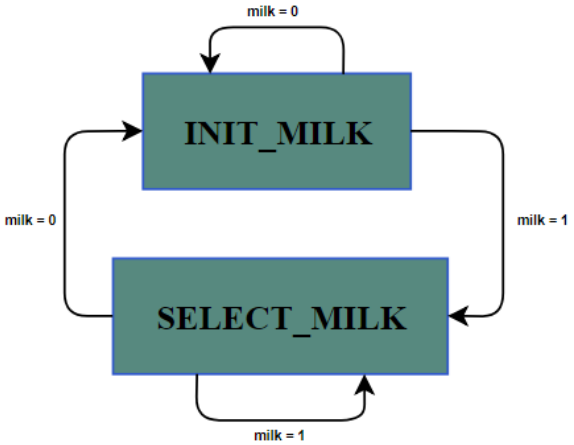
En este diagrama de estados podemos ver la secuencia que siguen los tres estados principales, los cuales guardaremos en la variable current_stage, siendo estos estados, apagado, standby y sirviendo. Para pasar de uno a otro, así como los valores de los bits en las respectivas señales para que esto ocurra.

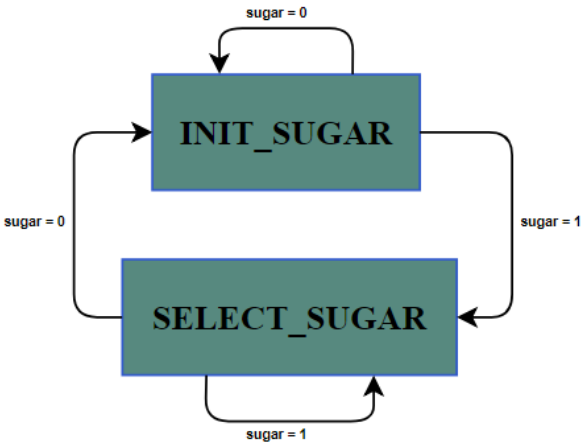


En el estado standby del diagrama anterior existen estados internos que se guardan en la variable coffee_time para elegir el tipo de café, dependiendo del tipo seleccionado, se cargará un tiempo diferente en el contador count. En los estados inicio y sirviendo (en este estado cuando ok esté activo) del diagrama anterior en la variable coffee_time se le asigna el valor de select_cafe.



En estos últimos diagramas se muestra como se pasa cambia entre sendos estados posibles para las variables de leche y azúcar. Un primer estado init donde se inicializan las variables internar y un segundo estado donde se seleccionan las distintas opciones de estas variables.

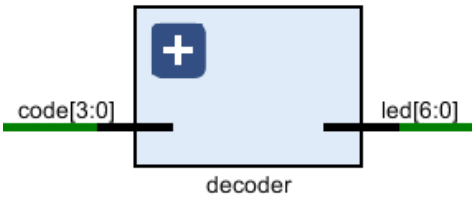




3.5. Decodificador

3.5.1. Diagrama Decodificador

Disponemos de ocho instanciamientos de módulos como este, uno por cada display. Cada uno con su respectiva entrada provenientes del módulo anterior, la máquina de estados. Un instanciamiento del módulo se usará para las unidades y otro para las decenas, así también se usarán uno para la letra L de leche, otra para la temperatura de la leche y cuatro para el nivel de azúcar. La salida LED de este módulo irá al módulo refresco de displays con la nueva codificación.



Este módulo se encarga de asignar a cada valor, dependiendo del decodificador, en código binario (4 dígitos) un vector de valor también en código binario, pero de mayor tamaño (7 dígitos). Cada binario de este nuevo código corresponde a un segmento LED de los displays finales (7 segmentos LED), según la combinación asignada podremos encender los LEDs apropiados para ver en el display un valor en BCD, una letra, un guion o todo apagado.

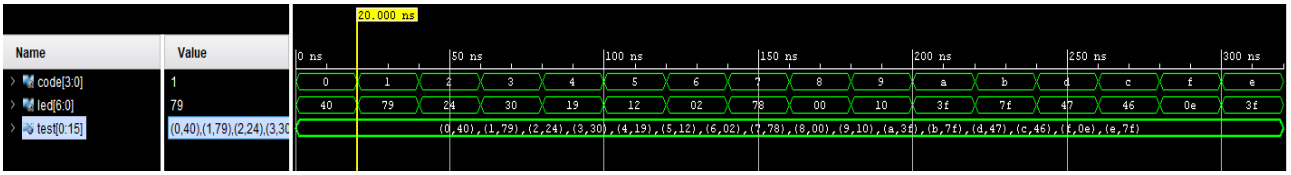
3.5.2. Código Decodificador

Los diez primeros códigos representan los números decimales para realizar la cuenta atrás para los diferentes tipos de café. Las cinco últimas salidas del decodificador encienden los leds de los displays de forma que, si están todos apagados se usa para los displays que deban estar así, el guion (-) para la espera de elegir temperatura de la leche, la L designa Leche, la C para Caliente y la F para Fría.

```
entity decoder is
    PORT( code : in STD_LOGIC_VECTOR (3 downto 0);
          led : out STD_LOGIC_VECTOR (6 downto 0)
          );
end decoder;

architecture Behavioral of decoder is

    signal segmento : STD_LOGIC_VECTOR (6 downto 0);

BEGIN

    with code select
        -- GFEDCBA LED Order
        segmento <=  "1000000" WHEN "0000", -- "0"
                     "1111001" WHEN "0001", -- "1"
                     "0100100" WHEN "0010", -- "2"
                     "0110000" WHEN "0011", -- "3"
                     "0011001" WHEN "0100", -- "4"
                     "0010010" WHEN "0101", -- "5"
                     "0000010" WHEN "0110", -- "6"
                     "1111000" WHEN "0111", -- "7"
                     "0000000" WHEN "1000", -- "8"
                     "0010000" WHEN "1001", -- "9"
                     "0111111" WHEN "1010", -- "-"
                     "1111111" WHEN "1011", -- " "
                     "1000111" WHEN "1101", -- "L"
                     "1000110" WHEN "1100", -- "C"
                     "0001110" WHEN "1111", -- "F"
                     "0111111" WHEN OTHERS;

        led <= segmento;

end Behavioral;
```

3.5.3. Simulación Decodificador

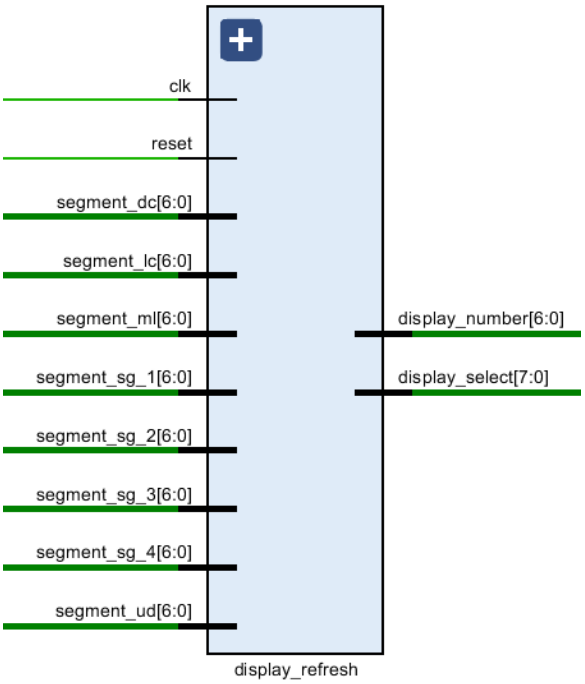
En esta simulación se hace un test que recorre todos posibles valores de salida del decodificador, probando todos los binarios de entrada.



3.6. Refresco de Displays

3.6.1. Diagrama Refresco de Displays

Como último módulo tenemos el encargado de permitirnos ver los displays de manera natural, ya que, no podemos activar todos los displays al mismo tiempo. En el siguiente código haremos que ambos displays se iluminen de manera alterna, pero a una frecuencia tan rápida que para el ojo humano no es perceptible, así veremos ambos números al mismo tiempo, utilizando la entrada clk a 400Hz que nos entrega el instanciamiento del módulo divisor de frecuencia.



Este módulo consta de diez entradas, una para el clk y otra para el reset, las otras ocho son sedment_ud y segment_dc, segment_ml, segment_lc, segment_sg_1, segment_sg_2, segment_sg_3 y segment_sg_4, cuyos valores vienen de las salidas de los módulos decodificadores. Dependiendo del nivel al que esté el reloj de este módulo se iluminará un display u otro. Usando un código binario de 4 dígitos enviaremos a los ánodos de los dos displays un 1 o un 0 para activarlos o desactivarlos (display activo en 0).

A la salida del módulo, finalmente, se enviarán los valos de salida hacia el módulo cafetera y de ahí al usuario. Las salidas son display_number, el cual contine el valor del display y display_select, el display elegido. En los displays podremos ver la cuenta atrás para servir el tipo de café seleccionado por el usuario de la máquina de café, si quiere leche, la temperatura de esta y el nivel de azúcar.

3.6.2. Código Refresco de Displays

Como se puede ver en el código, se usa un contador llamado refresh_counter para recorrer los ocho displays de uno en uno, e ir encendiéndolos a una velocidad de 400 Hz, imperceptible para el ojo humano. Además, se dispone de una condición al principio, en la que, si el botón reset está activado, mandaremos a todos los displays la señal de todos los segmentos apagados y todos los dos ánodos de los displays a 1, es decir, apagados.

```
variable refresh_counter : integer range 0 to 7;

BEGIN

    display_select <= "11111111"; -- para quitar latch
    display_number <= "1111111"; -- para quitar latch

    if reset = '1' then
        display_select <= segment_select_reset; -- displays apagados
        display_number <= segment_number_reset; -- sin numero
        refresh_counter := 0;

    elsif clk = '1' and clk'event then
        if refresh_counter = 7 then
            refresh_counter := 0;
        else
            refresh_counter := refresh_counter + 1;
        end if;

        case refresh_counter is

                WHEN 0 => display_select <= "11111110";
                          display_number <= segment_ud; -- unidades

                WHEN 1 => display_select <= "11111101";
                          display_number <= segment_dc; -- decenas

                WHEN 2 => display_select <= "11111011";
                          display_number <= segment_ml; -- leche temperatura

                WHEN 3 => display_select <= "11110111";
                          display_number <= segment_lc; -- letra L de leche

                WHEN 4 => display_select <= "11101111";
                          display_number <= segment_sg_1; -- azucar nivel 1

                WHEN 5 => display_select <= "11011111";
                          display_number <= segment_sg_2; -- azucar nivel 2

                WHEN 6 => display_select <= "10111111";
                          display_number <= segment_sg_3; -- azucar nivel 3

                WHEN 7 => display_select <= "01111111";
                          display_number <= segment_sg_4; -- azucar nivel 4

                WHEN OTHERS => NULL;

        end case;
    end if;
end process;
```

3.6.3. Simulación Refresco de Displays

Se ha cargado en los displays en orden de derecha a izquierda los valores de “9”, “1”, “F”, “L”, “1”, “1” y los dos últimos sin número. En pantalla se podrá ver un tiempo de 19 segundos, leche fría representada por LF y dos niveles de azúcar mostrados por los dos “1” del final. La simulación del refresco de displays muestra cómo se van encendiendo los diferentes displays alternadamente, varían el valor en display_select.