CONTROL PROGRAMABLE DE LAVADORA

1. ESPECIFICACIONES

Sobre la EMP7128SLC84-6, se quiere diseñar un circuito secuencial síncrono, que sirva como controlador de una lavadora sencilla, la cual estará formada por tres ciclos de funcionamiento (lavado, aclarado y centrifugado) de la misma duración temporal. El circuito dispondrá de una señal de reloj CLK sensible a flanco de subida, una señal de reset RST asíncrona y activa a nivel alto e incluirá dos bloques jerárquicos, tal como se puede apreciar en la figura 1.

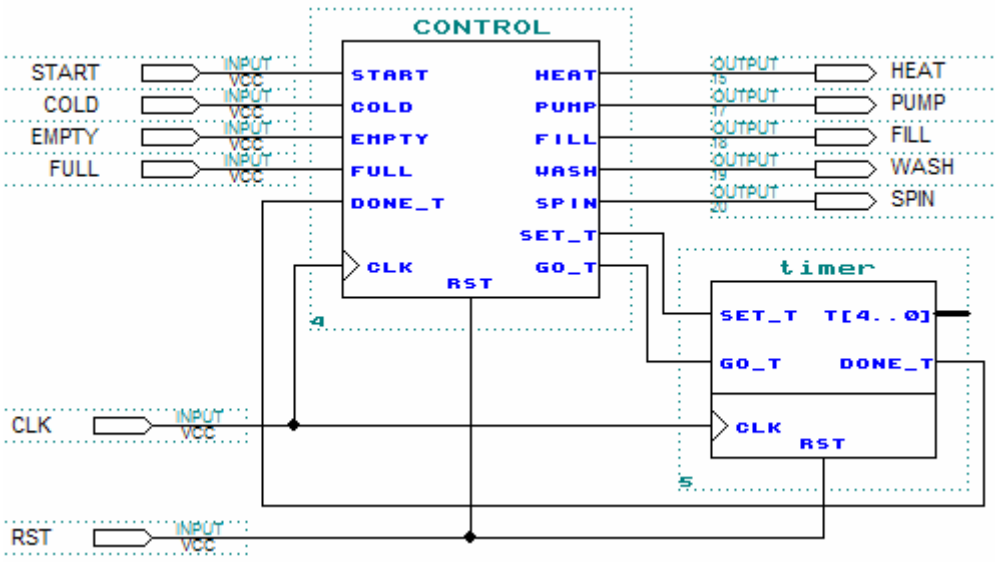

Figura 1. Controlador de una lavadora sencilla.

El bloque Temporizador o Timer, estará constituido por un contador descendente (usar una lpm_counter) de 5 bits T[4..0] que funcionará de la siguiente manera:

- Al iniciar cualquiera de los 3 ciclos, cargará síncronamente (al recibir un pulso Set_T desde Control) o asíncronamiente (tras RST global) un mimo valor predeterminado por el diseñador.
- De uno en uno, descontará (Go_T a '1' desde Control) esta cifra, hasta alcanzar el '0', generando una señal pulso Done_T, que indicará a control el fin del ciclo en cuestión.

El bloque Controlador o Control, es una máquina de estados finitos con las siguientes entradas y salidas:

- Entradas:
  - RST y CLK: Señales asíncronas de control comunes a Timer.
  - START: Un pulso a '0' iniciará el funcionamiento.
  - COLD: Se requiere calentar el agua.
  - EMPTY: Tambor de lavado vacío de agua.
  - FULL: Tambor de lavado lleno de agua.
  - Done_T: El temporizador alcanzó el fin (el cero).
- Salidas:
  - HEAT: Activa el calefactor de agua.
  - PUMP: Activa la bomba del agua (llenado o vaciado).
  - FILL: A '1' implica llenado, a '0' vaciado, del tambor.
  - SPIN: Acciona el motor para centrifugado.

- Set_T: Pulso de carga del timer.
- Go_T: Señal de descuento del timer.

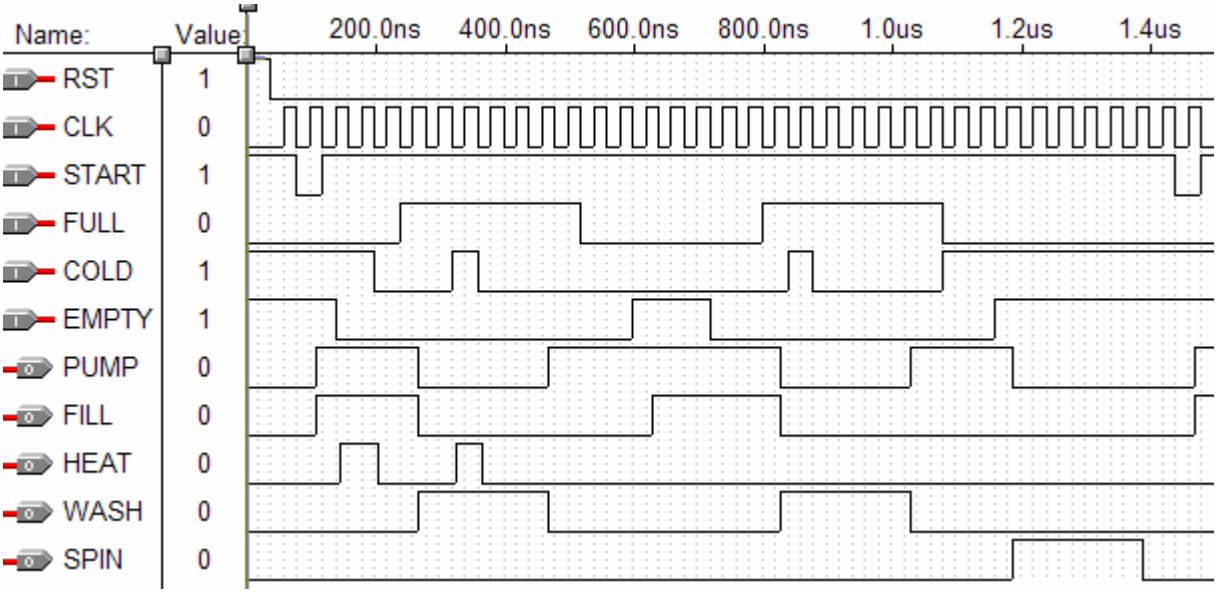

Figura 2. Diagrama temporal de comportamiento de las señales.

2. DESCRIPCIÓN DEL FUNCIONAMIENTO

Tras RST, cuando el controlador reciba por la señal START un pulso '0' (START estará normalmente a '1' en todo el funcionamiento, si START sufriera un pulso '0' en otro estado distinto del inicial, se retornará al estado inicial), la máquina se llenará de agua y la calentará hasta alcanzar el nivel y la temperatura adecuados (estado Fill1). La lavadora puede partir de un tambor no vacío de agua, pero si está vacío no debe actuar el calefactor (se quemaría).

Entonces se iniciará la operación de lavado que seguirá hasta que el temporizador alcance el cero manteniendo la temperatura establecida (estado Washing). Después se vaciará el agua sucia y se rellenará, de nuevo, con agua fría (estados Empty1 y Fill2). Se aclarará en frío siguiendo el mismo accionamiento del motor que en el lavado, hasta que el temporizador alcance, otra vez, el cero (estado Clearing). Por último, tras vaciar de nuevo el tambor (Empty2), se iniciará el centrifugado (Spinning) que finalizará al alcanzar cero el temporizador devolviendo al controlador al estado inicial. El funcionamiento del sistema viene especificado por la máquina de estados de la figura 3.

3. ORGANIZACIÓN DE TAREAS

1. Diseño del bloque Timer
    a. Implementar y compilar desde Captura de Esquemas
    b. Verificar el bloque Timer
2. Diseño del bloque Control
    a. Implementar y compilar desde descripción VHDL
    b. Verificar el bloque Control
3. Diseño de la jerarquía superior
    a. Implementar y compilar desde captura de esquemas.
    b. Verificar el bloque TODO
4. Rellenar

   a. Número de LCs utilizadas
   b. Fmax de operación
5. Realizar las siguientes asignaciones
   a. RST como Global Clear
   b. START como Entrada Dedicada

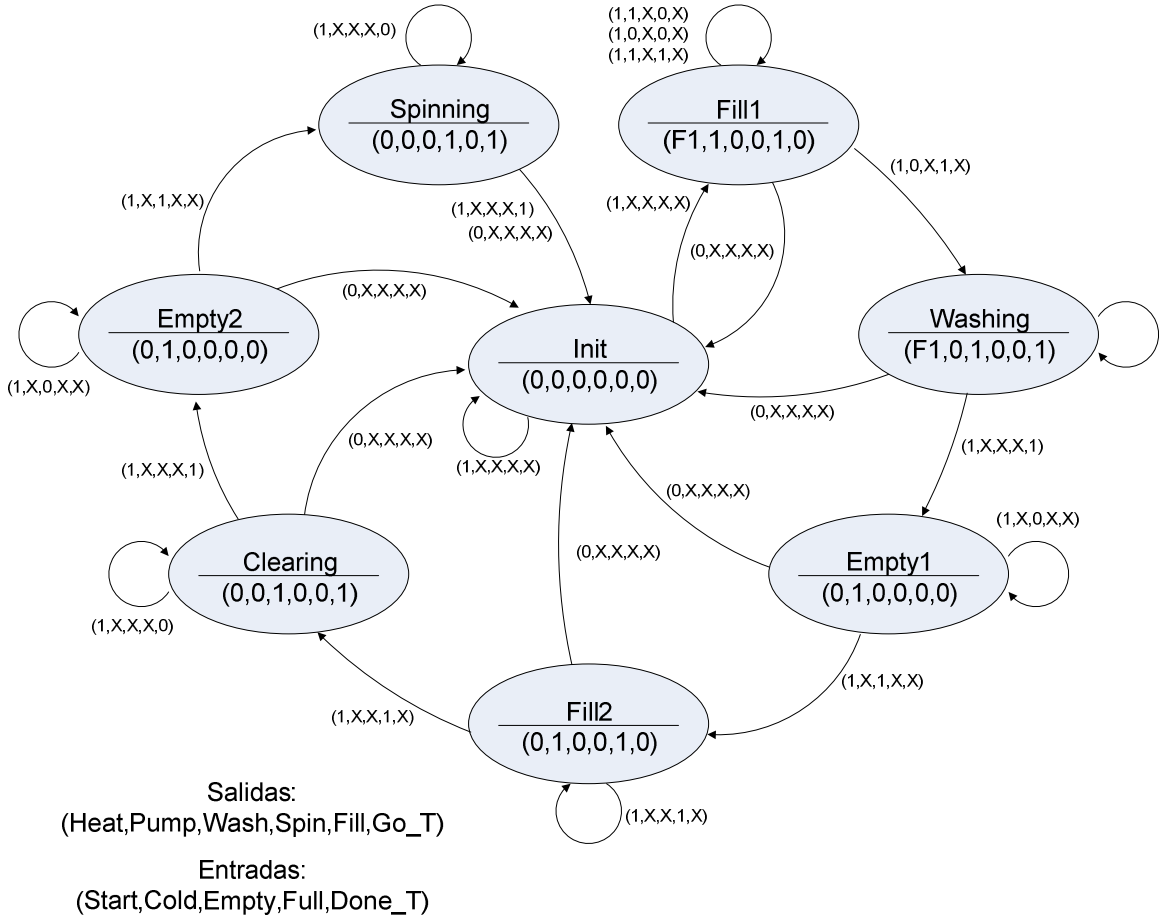

Figura 3. Máquina de estados del control programable de lavadora.

4. SOLUCIÓN DEL EJERCICIO

4.1. DISEÑO DEL BLOQUE TIMER

El esquema del bloque Timer se puede apreciar en la figura 4, donde cabe destacar los siguientes aspectos:

- El empleo de Wire para redenominar una señal
- El uso en la lpm_counter de:
  - EQ[0], para detectar el paso por '0'
  - ASET, para cargar el valor, con RST asíncrono, predeterminado LPM_AVALUE (igual a LPM_SVALUE) de modo que configure la misma duración que el síncrono por SET_T.
  - LPM_SVALUE es pequeño para evitar una larga simulación (por ejemplo, 3).

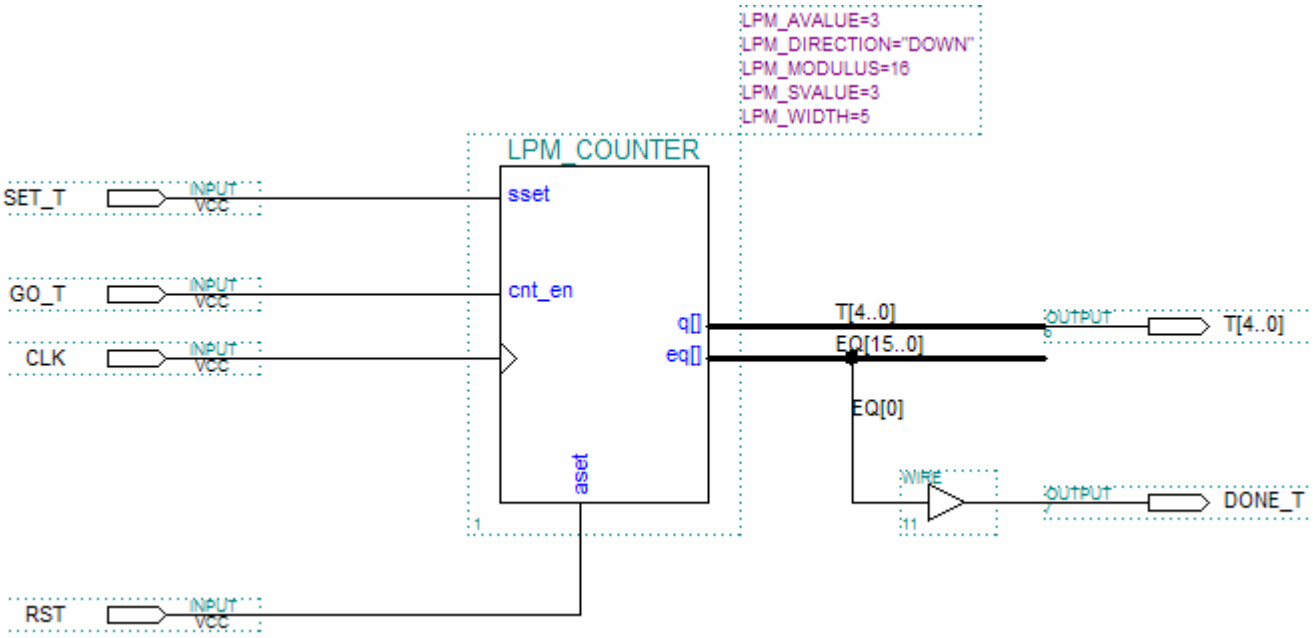

Figura 4. Esquemático del bloque Timer

La verificación (figura 5) habrá de poner de manifiesto tanto el correcto diseño del Esquema como el comportamiento temporal de las señales de interfase con Control: SET_T, GO_T y DONE_T. Observe que no importa demasiado si SET_T y GO_T se activan simultánea o secuencialmente.

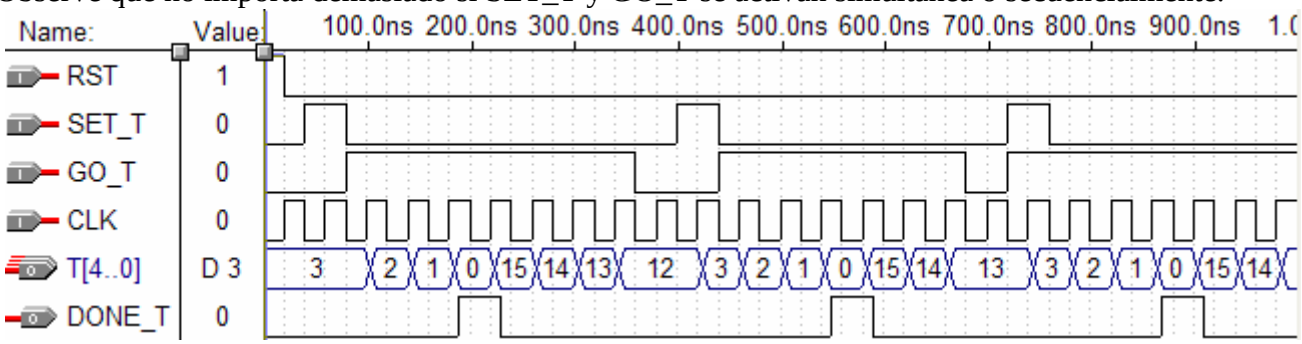

Figura 5. Verificación temporal del funcionamiento del bloque Timer

4.2. DISEÑO DEL BLOQUE CONTROL

Para diseñar esta unidad recomiendo el empleo de ModelSim ya que la herramienta de Help es mejor que en el caso de MaxPlus. Una vez diseñado, exportarlo a este último. La descripción VHDL que se ha empleado para Control es la siguiente:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY control IS
        PORT
        (       RST     : IN STD_LOGIC;
                CLK     : IN STD_LOGIC;
                START   : IN STD_LOGIC;
                COLD    : IN STD_LOGIC;
                EMPTY   : IN STD_LOGIC;
                FULL    : IN STD_LOGIC;
                DONE_T  : IN STD_LOGIC;
                HEAT    : OUT STD_LOGIC;
                PUMP    : OUT STD_LOGIC;
                FILL    : OUT STD_LOGIC;
                WASH    : OUT STD_LOGIC;
                SPIN    : OUT STD_LOGIC;
```

```
                SET_T     : OUT STD_LOGIC;
                GO_T      : OUT STD_LOGIC
        );
END control;

ARCHITECTURE sed OF control IS
        TYPE mystate is ( Init, Fill1, Washing, Empty1, Fill2, Clearing, Empty2, Spinning );
        ATTRIBUTE ENCODING : STRING;
        ATTRIBUTE ENCODING OF mystate: TYPE IS "000 101 111 011 100 110 010 001";

-- Codificación empleada para la FSM: 2 Procesos.

        SIGNAL present_state, next_state: mystate;

BEGIN
        Transition_T: PROCESS (present_state, START)
        BEGIN
                CASE present_state IS
                        WHEN Init =>
                                HEAT      <= '0';
                                PUMP      <= '0';
                                WASH      <= '0';
                                SPIN      <= '0';
                                FILL      <= '0';
                                GO_T      <= '0';
                                IF ( START = '0' ) THEN
                                        next_state <= Fill1;
                                ELSE
                                        next_state <= Init;
                                END IF;
                        WHEN Fill1 =>
                                HEAT      <= NOT( EMPTY ) AND COLD;
                                PUMP      <= '1';
                                WASH      <= '0';
                                SPIN      <= '0';
                                FILL      <= '1';
                                GO_T      <= '0';
                                IF (START = '1') THEN
                                        IF (FULL ='1' AND COLD = '0') THEN
                                           next_state          <= Washing;
                                        ELSE
                                           next_state          <= Fill1;
                                        END IF;
                                ELSE
                                        next_state <= Init;
                                END IF;
                        WHEN Washing =>
                                HEAT      <= NOT (EMPTY) AND COLD;
                                PUMP      <= '0';
                                WASH      <= '1';
                                SPIN      <= '0';
                                FILL      <= '0';
                                GO_T      <= '1';
                                IF (START = '1') THEN
                                        IF (DONE_T ='1') THEN
                                           next_state          <= Empty1;
                                        ELSE
                                           next_state          <= Washing;
                                        END IF;
                                ELSE
                                        next_state <= Init;
                                END IF;
                        When Empty1 =>
                                HEAT      <= '0';
                                PUMP      <= '1';
                                WASH      <= '0';
                                SPIN      <= '0';
                                FILL      <= '0';
                                GO_T      <= '0';
                                IF (START = '1') THEN
                                        IF (EMPTY ='1') THEN
                                           next_state          <= Fill2;
                                        ELSE
                                           next_state          <= Empty1;
                                        END IF;
                                ELSE
```

```
                        next_state <= Init;
                END IF;
        WHEN Fill2 =>
                HEAT    <= '0';
                PUMP    <= '1';
                WASH    <= '0';
                SPIN    <= '0';
                FILL    <= '1';
                GO_T    <= '0';
                IF (START = '1') THEN
                        IF (FULL ='1') THEN
                           next_state       <= Clearing;
                        ELSE
                           next_state       <= Fill2;
                        END IF;
                ELSE
                        next_state <= Init;
                END IF;
        WHEN Clearing =>
                HEAT    <= '0';
                PUMP    <= '0';
                WASH    <= '1';
                SPIN    <= '0';
                FILL    <= '0';
                GO_T    <= '1';
                IF (START = '1') THEN
                        IF (DONE_T ='1') THEN
                           next_state       <= Empty2;
                        ELSE
                           next_state       <= Clearing;
                        END IF;
                ELSE
                        next_state <= Init;
                END IF;
        When Empty2 =>
                HEAT    <= '0';
                PUMP    <= '1';
                WASH    <= '0';
                SPIN    <= '0';
                FILL    <= '0';
                GO_T    <= '0';
                IF (START = '1') THEN
                        IF (EMPTY ='1') THEN
                           next_state       <= Spinning;
                        ELSE
                           next_state       <= Empty2;
                        END IF;
                ELSE
                        next_state <= Init;
                END IF;
        WHEN Spinning =>
                HEAT    <= '0';
                PUMP    <= '0';
                WASH    <= '0';
                SPIN    <= '1';
                FILL    <= '0';
                GO_T    <= '1';
                IF (START = '1') THEN
                        IF (DONE_T ='1') THEN
                           next_state       <= Init;
                        ELSE
                           next_state       <= Spinning;
                        END IF;
                ELSE
                        next_state <= Init;
                END IF;
                -- como hay 8 estados no me preocupan OTHERS
        END CASE;
END PROCESS;

Transition_P: PROCESS (CLK, RST)
BEGIN
        IF RST= '1' THEN
                present_state <= INIT;
        ELSIF CLK'EVENT AND CLK = '1' THEN
                present_state <= next_state;
```

```
                END IF;
        END PROCESS;

        Gen_Set_T: PROCESS (CLK, RST)
        BEGIN
                IF RST = '1' THEN
                                SET_T <= '0';
                ELSIF CLK'EVENT AND CLK = '1' THEN
                                IF ( FULL = '1') THEN
                                        IF ( present_state = Fill1 AND COLD = '0') OR ( present_state = Fill2 ) THEN
                                                        SET_T     <= '1';
                                        ELSE
                                                        SET_T     <= '0';
                                        END IF;
                                ELSIF  ( present_state = Empty2 AND EMPTY = '1') THEN
                                                        SET_T     <= '1';
                                ELSE
                                                        SET_T     <= '0';

                                END IF;
                END IF;
        END PROCESS;
END sed;
```

El proceso siguiente describe la generación del Pulso SET_T. Observe que SET_T podría haber sido definida como Salida de los Estados previos a Washing, Clearing y Spinnig, con lo que habríamos estado cargando el temporizador continuamente, antes de efectuar estos estados. Se ha preferido la solución de generar un solo pulso de 1 Ciclo de CLK y describirla en un proceso distinto al anterior, por otro lado innecesario, por claridad. Observe que mientras HEAT, PUMP,..., GO_T son salidas Moore Directas, SET_T es una salida Mealy registrada para asegurar su integridad.

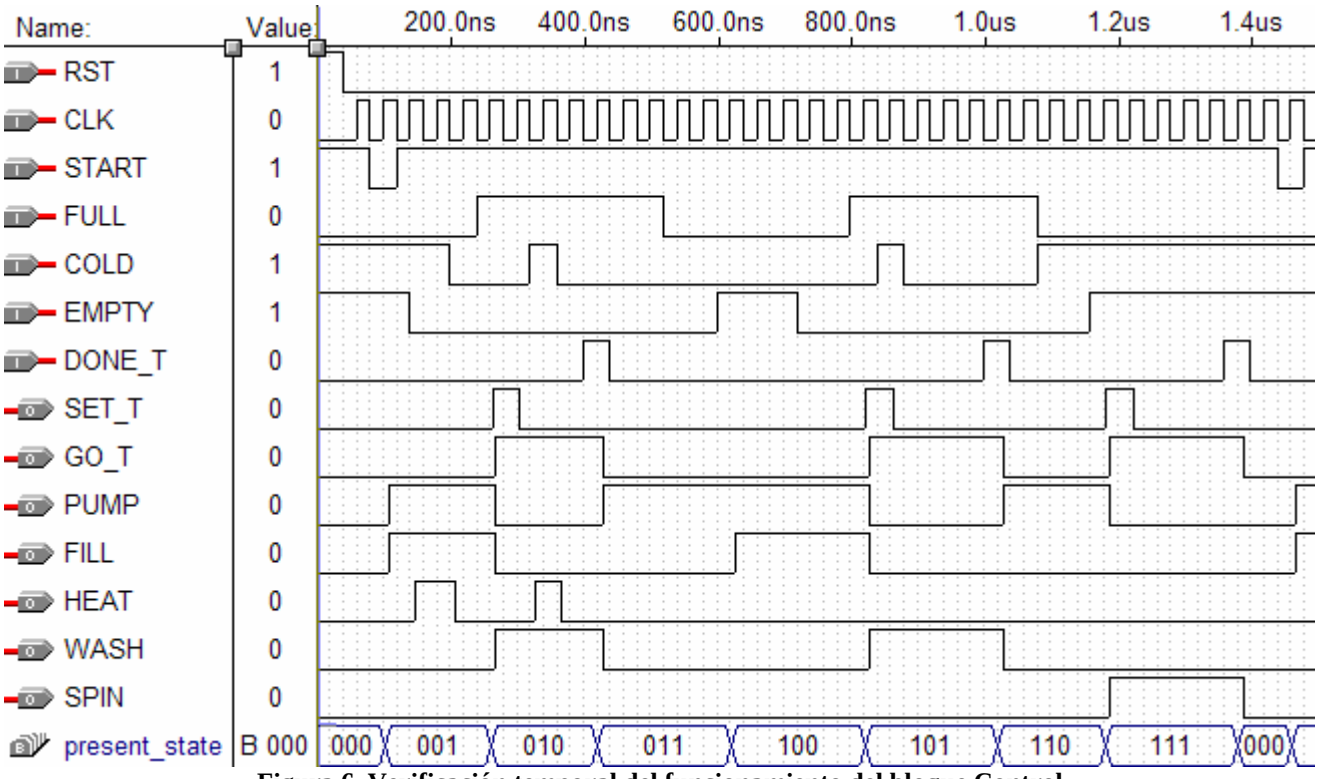

Figura 6. Verificación temporal del funcionamiento del bloque Control

4.3. DISEÑO DE LA JERARQUÍA SUPERIOR

Simplemente se realizan las conexiones de los dos bloques anteriores tal como se muestra en la figura 1. En la figura 2 se muestra el resultado de la simulación.

4.4. PRESTACIONES DEL DISEÑO

Examinando el archivo de report correspondiente al diseño y realizando un análisis temporal con la opción Registered Performance, se obtiene:

- Número de LCs utilizadas: 21.
- Fmax de operación: 56.49 MHz.

4.5. ASIGNACIONES

Para asignar RST y START, se emplea la herramienta FoorPlan Edit, activando Assign/Back-Annotate Project y cargar Layout/Current Assignments Floorplan, donde arrastramos RST y START a los pines dedicados tras lo que se compila nuevamente, observando en Layout/Last Compilation FloorPlan que RST y START se encuentran donde deben estar.