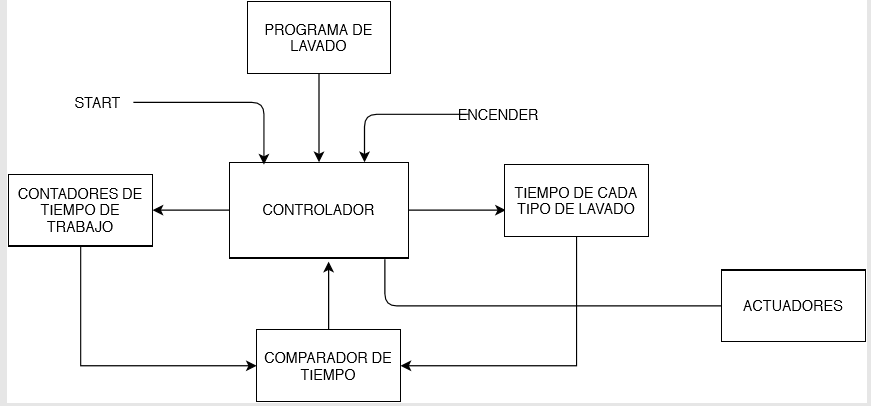
**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN**

**Nombres:** Joe Alvia Alvarado / María Fernanda Mora

**LAVADORA AUTOMATICA.**

1. **Explicación detallada:** Se presenta dos posibles soluciones a los requerimientos pedidos para la lavadora automática, del cual se ha decidido optar por la primera propuesta ya que cuenta con menos estados en el diagrama ASM, su método de validación es más preciso y eficiente que la propuesta número 2.
2. **Diagrama de bloques**



El diagrama de bloque mostrado muestra el comportamiento del sistema a implementar, Cuando el usuario presiona el botón ENCENDER, la lavadora esta lista para funcionar entonces solo se le debe configurar el PROGRAMA DE LAVADO mediante los botones de DELICADO, NORMAL, TOALLAS, JEAN además del botón de REMOJO.

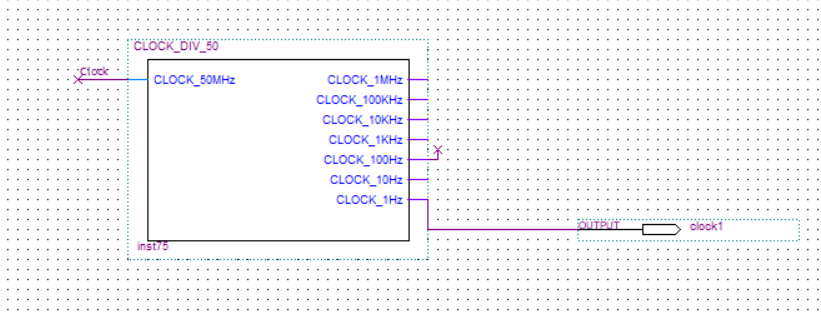
Una ve programada la lavadora se procede a presionar STAR iniciando el llenado del tanque y realizando comparaciones hasta llegar al valor designado.

Cuando todas las condiciones han sido seleccionadas, el tanque esta lleno entonces se procede a realizar las acciones de LAVADO, EXPRIMIDO, ENJUAGUE Y EXPRIMIDO FINAL tantas veces según el programa lo especifique.

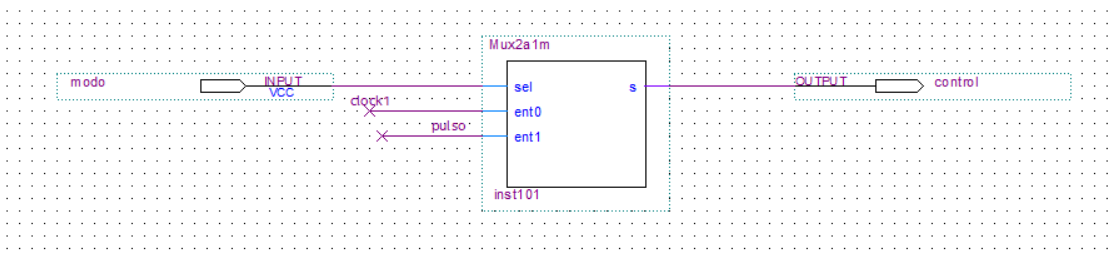
El bloque CONTROLADOR permite el intercambio entre los datos entre el tiempo de ejecución de la lavadora y el tiempo designado para cada operación, según el tiempo se procederá a la siguiente ejecución del programa hasta terminar con el DESAGUE.

1. **Partición Funcional**

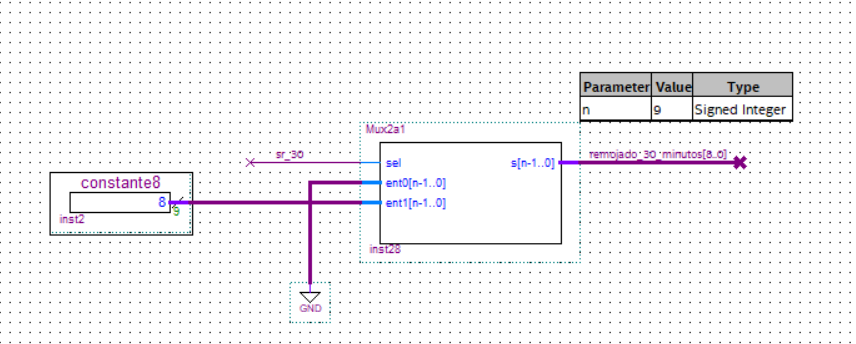
Se tiene un divisor de frecuencia de 1Hz con el que trabaja todo el sistema, no obstante, esta señal entra a un MUX de 2 a 1 en el que se podrá escoger el modo en que se quiere trabajar, es decir con: clock manual o clock de la FPGA. A continuación, se muestra el bloque:



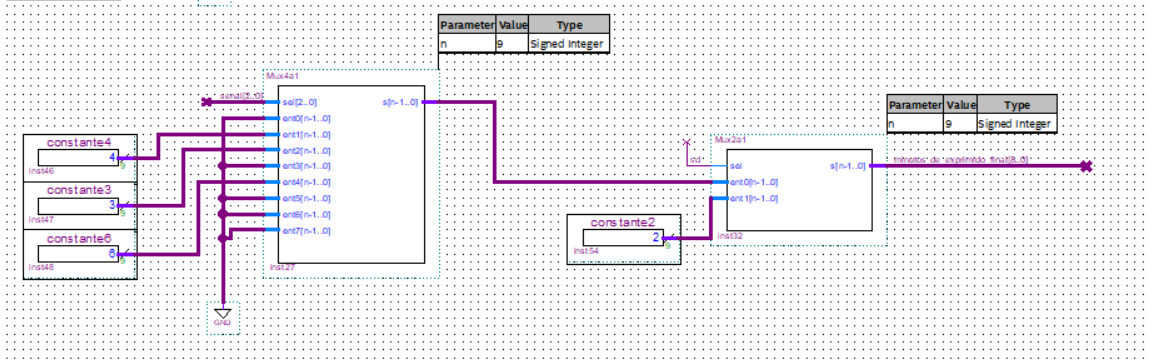
La salida de control entra a todos los bloques de la partición funcional, excepto en el convertidor analógico-digital en el que siempre cuenta con el clock del sistema.



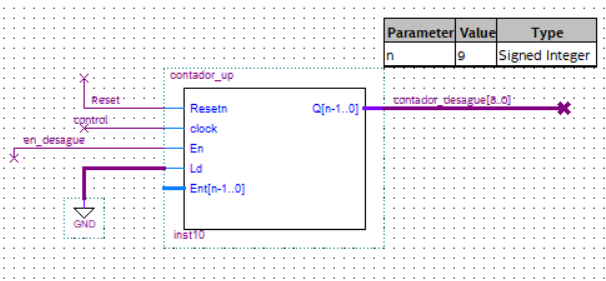
La señal sr\_30 habilita la opción de remojo o no remojo, de tal forma que se agrega o no 30 min a la opción de lavado.

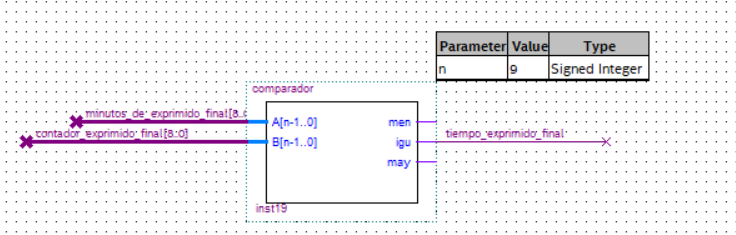


Estos bloques representan la selección de tiempo de acuerdo con el programa escogido, para lo cual se tiene las señales habilitadoras: sd,st,sj,sn.

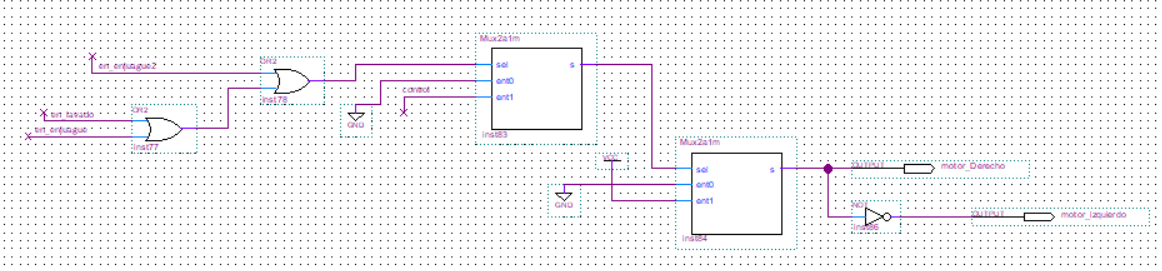


El contador up es el encargado de contar el tiempo que demora cada proceso y sera habilitado en cadena de acuerdo con la etapa de lavado en la que me encuentro, iniciando en: Lavado, Exprimido1, Enjuague1, Exprimido2, Enjuague2, ExprimidoFinal.

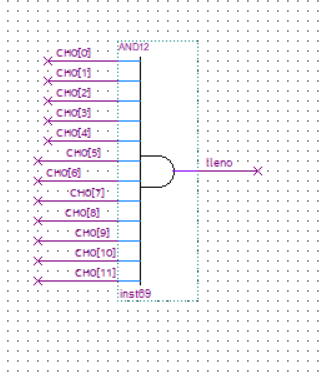


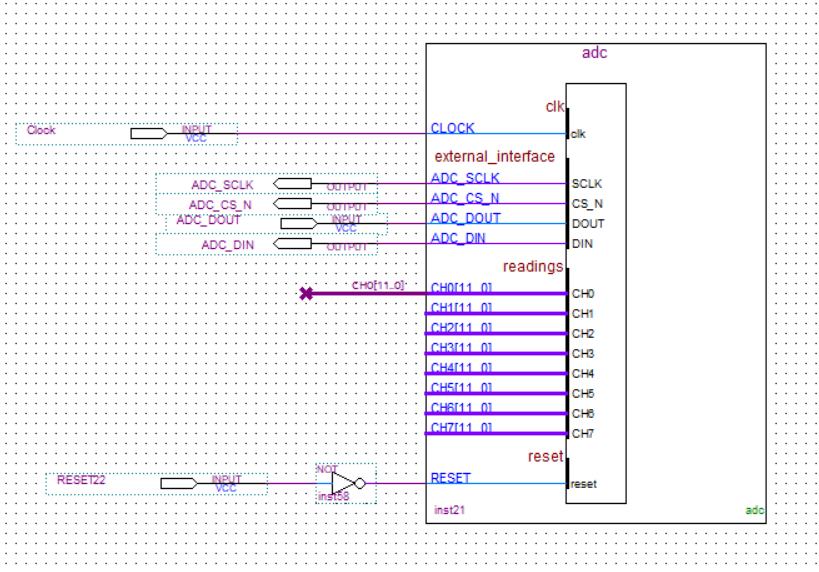
El comparador estará contrastando constantemente el tiempo fijo de cada lavado con la duración que esta teniendo la etapa de lavado que se está ejecutando en ese momento, de tal manera que marca un alto cuando son iguales. 

Este conjunto de MUX de 2 a 1 representan la habilitación o no de los motores derecho e izquierdo, los cuales se habilitan únicamente en la etapa de enjuague y lavado.

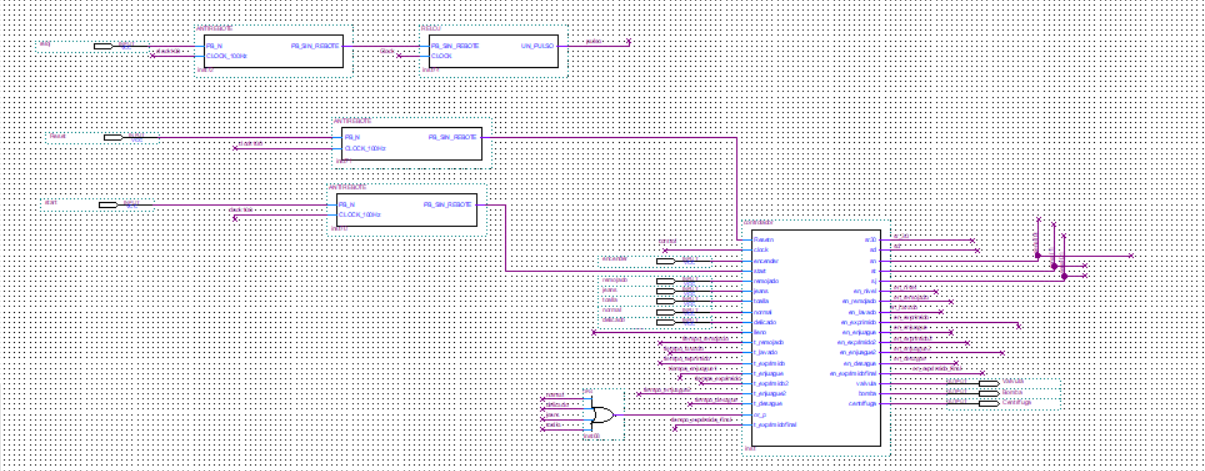


Se ha colocado una puerta OR12 a la salida del canal CHO del potenciómetro digital, el cual cuenta con 12 bits; con el objetivo de que cuando este se encuentre en su valor máximo, únicamente muestre un alto indicando que el nivel de agua del tanque esta lleno.



Este bloque representa al potenciómetro, en el cual se esta utilizando el puerto CH0 para transmitir los datos del potenciómetro analógico. 

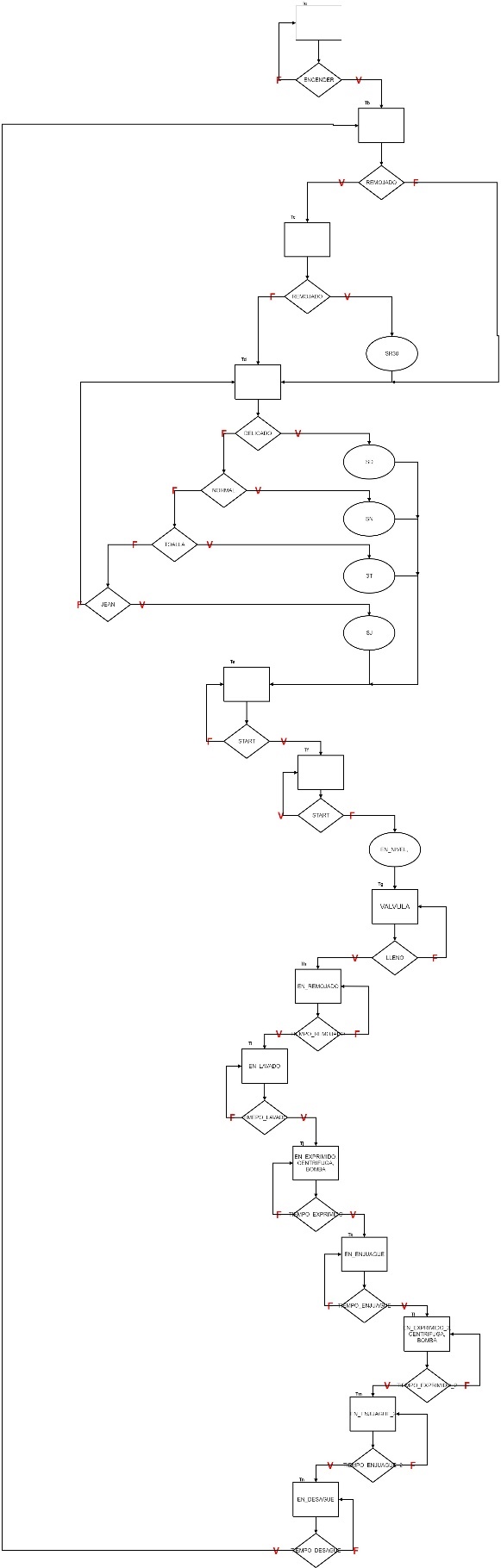
Este bloque representa al controlador con sus diferentes entradas y salidas que son utilizadas a lo largo de la partición funcional.



1. **Diagrama ASM del Controlador**

**OPCION #1.**

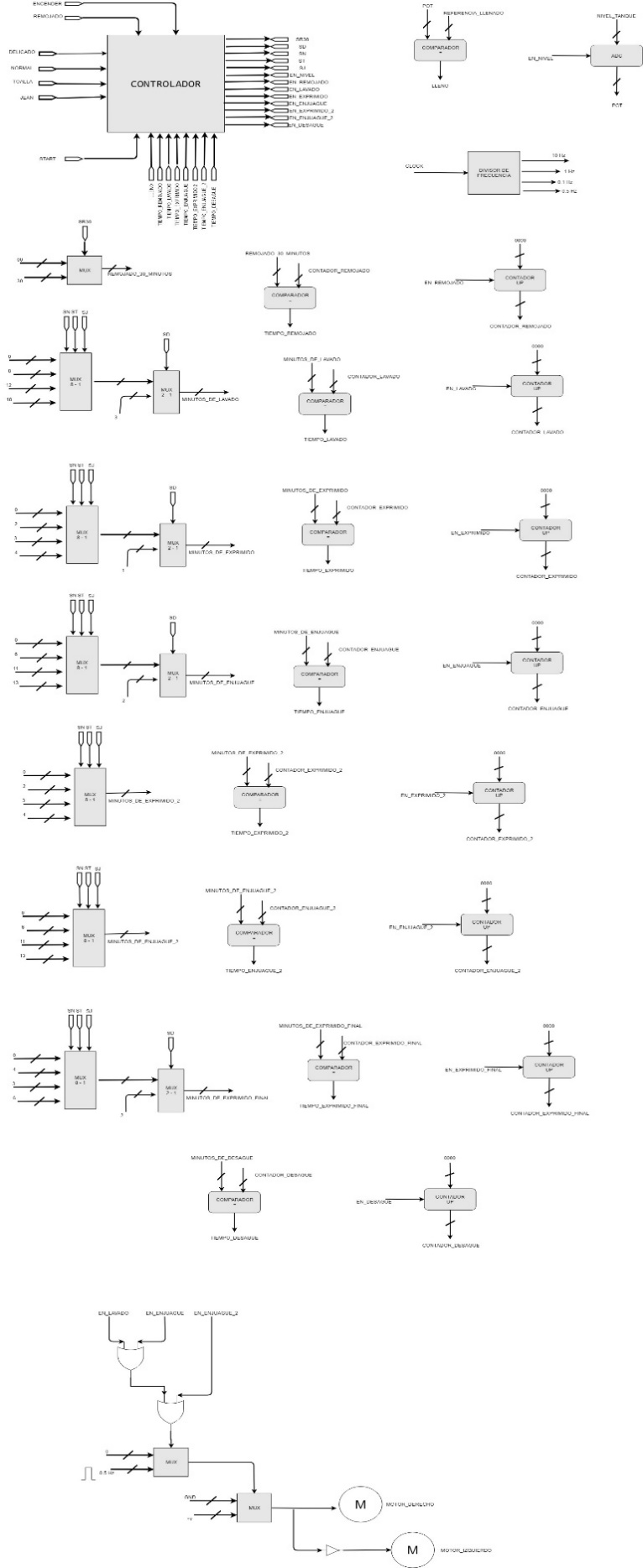
**DIAGRAMA ASM DEL CONTROLADOR.**



El diagrama ASM realizado permite comprender el funcionamiento y operación del sistema digital creado para la lavadora automática, la cual corresponde al siguiente comportamiento:

* En el Estado inicial la maquina no puede operar, es necesario presionar el botón ENCENDER para que esté en condiciones de recibir el tipo de programa y empezar las acciones.
* En el siguiente estado tenemos la opción de añadir 30 minutos al tiempo total de lavado mediante el botón REMOJDA, es necesario presionar y soltar el botón para validar los 30 minutos. En caso de validar los 30 minutos adicionales de remojo se enciende la señal SR30 la cual será la señal selectora de un MUX escogiendo entre los valores de 0 minutos y 30 minutos.
* Después de escoger o no la opción de REMOJADO continuamos a configurar el tipo de programa de lavado, escogiendo entre las opciones de DELICADO, NORMAL, TOALLA, JEAN mediante pulsadores los cuales activaran las señales SD, SN, ST, SJ respectivamente según sea el programa escogido. Estas señales serán las selectoras de varios MUX los cuales están configurados para arrojar el tiempo LAVADO, REMOJADO, ENJUAGADO, EXPRIMIDO, ECPRIMIDO FINAL según cada programa.
* Una vez escogido el programa podemos proceder a la acción de lavar, mediante la botonera STAR, pulsando y soltando, se activa la señal EN\_NIVEL que permitirá conocer el nivel de llenado del tanque hasta llegar al punto limite donde LLENO será verdadera y por lo tanto la lavadora podrá empezar a trabajar.
* Una vez lleno el tanque se procede a activar la señal de EN\_REMOJADO en caso de haber escogido la opción de remojado caso contrario para a la señal de EN\_LAVADO la cual activa el contador de lavador configurado al tiempo del programa escogido, una ves que este tiempo se haya concluido la señal TIEMPO\_LAVADO pasara a ser verdadera. Así el sistema activara los contadores según se lave, enjuague, exprima, hasta llegar al final donde la lavadora debe eliminar el agua utilizada mediante el desagüe, esto activa la señal EN\_DESAGUE y se toma el tiempo necesario.
* Una vez eliminada toda el agua procede a volver al estado Tb donde se debe escoger el tipo de programa de lavado y repetir el procedimiento nuevamente.

**PARTICION FUNCIONAL**

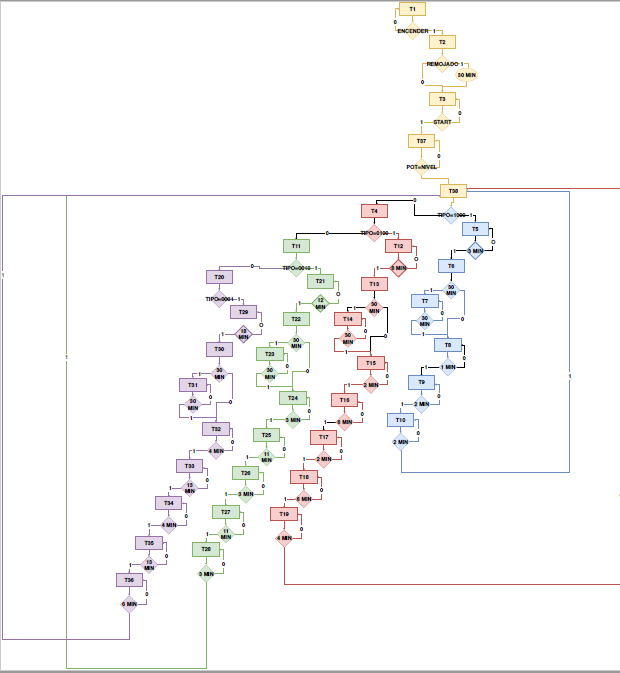


Las variables de la partición funcional son las siguiente:

* CLOCK: Reloj del sistema que se encarga de definir los pulsos de adquisición de datos del controlador.
* RESET: Señal que devuelve al primer estado de manera asincrónica.
* ENCENDER: permite encender la máquina para que esté listo para que podamos ingresar las configuraciones y datos.
* REMOJADO: tipo de configuración de la lavadora con un tiempo de lavado designado.
* DELICADO: tipo de configuración de la lavadora con un tiempo de lavado designado.
* NORMAL: tipo de configuración de la lavadora con un tiempo de lavado designado.
* TOALLA: tipo de configuración de la lavadora con un tiempo de lavado designado.
* JEAN: tipo de configuración de la lavadora con un tiempo de lavado designado.
* START: Inicia las operaciones de la lavadora.
* LLENO: Salida que permite conocer si el tanque está lleno.
* TIEMPO\_RMEOJADO: salida que permite conocer si el tiempo de remojo se ha concluido.
* TIEMPO\_LAVADO: salida que permite conocer si el tiempo de lavado se ha concluido.
* TIEMPO\_EXPRIMIDO: salida que permite conocer si el tiempo de exprimido se ha concluido.
* TIEMPO\_ENJUAGUE: salida que permite conocer si el tiempo de enjuague se ha concluido.
* TIEMPO\_EXPRIMIDO\_2: salida que permite conocer si el tiempo de exprimido se ha concluido.
* TIMEPO\_ENJUAGUE\_2: salida que permite conocer si el tiempo de enjuague se ha concluido.
* TIEMPO\_DESAGUE: salida que permite conocer si el tiempo de desagüe se ha concluido.
* SR30: señal que habilita 30 minutos más al tiempo de ejecución.
* SD: señal que habilita el tiempo de programa designado a DELICADO.
* SN: señal que habilita el tiempo de programa designado a NORMAL.
* ST: señal que habilita el tiempo de programa designado a TOALLA.
* SJ: señal que habilita el tiempo de programa designado a JEAN.
* EN\_NIVEL: señal habilitadora del contador que lleva el tiempo de nivel de llenado de tanque.
* EN\_REMOJADO: señal habilitadora del contador que lleva el tiempo de remojado.
* EN\_LAVADO: señal habilitadora del contador que lleva el tiempo de lavado.
* EN\_EXPRIMIDO: señal habilitadora del contador que lleva el tiempo de exprimido.
* EN\_ENJUAGUE: señal habilitadora del contador que lleva el tiempo de enjuague.
* EN\_EXPRIMIDO\_2: señal habilitadora del contador que lleva el tiempo de exprimido.
* EN\_ENJUAGUE\_2: señal habilitadora del contador que lleva el tiempo de enjuague.
* EN\_DESAGUE: señal habilitadora del contador que lleva el tiempo de desagüe.
* NIVEL\_TANQUE: señal analógica que mide el nivel de llenado del tanque.
* POT: señal digital que mide el nivel de llenado del tanque.
* REFERENCIA\_LLENADO: nivel al que se debe llenar el tanque

**OPCION #2.**

**DIAGRAMA ASM DEL CONTROLADOR.**

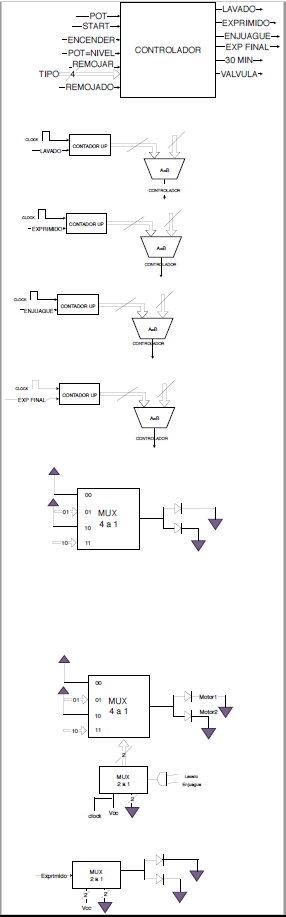


El diagrama ASM realizado permite comprender el funcionamiento y operación del sistema digital creado para la lavadora automática, la cual corresponde al siguiente comportamiento:

* En el estado T1 la maquina necesita que se presione el botón de encender para iniciar su funcionamiento, Luego en el siguiente estado se puede escoger entre la opción de añadir o no 30 minutos de remojado al programa mediante la botonera REMOJADO que activa la salida tipo Mealy MIN, posteriormente el sistema espera a que se presione el botón START para pasar al siguiente estado en que se verifica que el tanque de agua esta lleno mediante la señal de NIVEL, luego de esto escogemos entre los diferentes tipos de programa y cada uno con su diferente tiempo de lavado, exprimido, enjuague y todos retornan al estado T38 donde se procede nuevamente a escoger el tipo de programa de lavado.

Esta propuesta consta de varios estados por cada programa de lavado, reduciendo el numero de bloque en la partición funcional pero aumentado las variables del controlador.

**PARTICION FUNCIONAL**



1. **Código VHDL completo de la MSS**, usar un process para decodificador de estados siguiente–memoria de estados, y un process para el decodificador de salidas (con su respectiva explicación a detalle).

|  |
| --- |
| library ieee;  use ieee.std\_logic\_1164.all;  entity controlador is  Port(  Resetn, clock: in std\_logic;  encender, start, remojado, jeans, toalla, normal, delicado: in std\_logic;  lleno, t\_remojado, t\_lavado, t\_exprimido, t\_enjuague, t\_exprimido2, t\_enjuague2, t\_desague, or\_p, t\_exprimidofinal: in std\_logic;  sr30,sd,sn,st,sj,en\_nivel, en\_remojado, en\_lavado, en\_exprimido, en\_enjuague,en\_exprimido2,en\_enjuague2,en\_desague,en\_exprimidofinal, valvula, bomba, centrifuga: out std\_logic);    end controlador;  architecture solucion of controlador is  type estado is (ta,tb,td,te,tf,tg,th,ti,tj,tk,tl,tm,tn,tp);  signal y: estado;  begin  Process(Resetn, clock)  begin  if Resetn = '1' then y <= ta;  elsif clock'event and clock = '1' then  case y is  when ta=> if encender='1'then y<= tb; else y<= ta; end if;  when tb=> y<= td;  when td=> if or\_p = '1' then y<=te; else y<= td; end if;  when te=> if start='1' then y<= tf; else y<=te; end if;  when tf=> if start='1' then y<= tf; else y<=tg; end if;  when tg=> if lleno='1' then y<=th; else y<=tg; end if;  when th=> if t\_remojado='1' then y<=ti; else y<=th; end if;  when ti=> if t\_lavado='1' then y<=tj; else y<=ti; end if;  when tj=> if t\_exprimido='1'then y<=tk; else y<=tj; end if;  when tk=> if t\_enjuague='1' then y<=tl; else y<=tk; end if;  when tl=> if t\_exprimido2='1' then y<=tm; else y<=tl; end if;  when tm=> if t\_enjuague2='1' then y<=tn; else y<=tm; end if;  when tn=> if t\_exprimidofinal='1' then y<=tp; else y<=tn; end if;  when tp=> if t\_desague='1' then y<=ta; else y<=tp; end if;    end case;  end if;  end process;  process (y,remojado, delicado, normal, toalla, jeans, start)  begin  sr30<= '0'; sd<= '0';sn <= '0';st <= '0';sj <= '0';  en\_nivel<= '0'; en\_remojado<= '0'; en\_lavado<= '0';  bomba<= '0'; centrifuga<= '0'; en\_exprimido<= '0';  en\_enjuague<= '0';en\_exprimido2 <= '0';en\_enjuague2<= '0';  en\_desague<= '0'; valvula<= '0';en\_exprimidofinal<='0';  case y is  when ta=>  when tb=>sr30<= remojado;    when td=> sd<=delicado;sn<=normal;st<=toalla;sj<=jeans;sr30<= remojado;  when te=> sd<=delicado;sn<=normal;st<=toalla;sj<=jeans;sr30<= remojado;  when tf=> sr30<= remojado;sd<=delicado;sn<=normal;st<=toalla;sj<=jeans;en\_nivel<=not(start);  when tg=>  sd<=delicado;sn<=normal;st<=toalla;sj<=jeans;valvula<='1';sr30<=remojado;      when th=>  sd<=delicado;sn<=normal;st<=toalla;sj<=jeans;en\_remojado<='1';sr30<=remojado;    when ti=>  sd<=delicado;sn<=normal;st<=toalla;sj<=jeans;en\_lavado<='1';sr30<=remojado;      when tj=>  sd<=delicado;sn<=normal;st<=toalla;sj<=jeans;en\_exprimido<='1';centrifuga<='1'; bomba<='1';sr30<= remojado;    when tk=> sd<=delicado;sn<=normal;st<=toalla;sj<=jeans;en\_enjuague<='1';centrifuga<='0'; bomba<='0';sr30<= remojado;    when tl=> sd<=delicado;sn<=normal;st<=toalla;sj<=jeans;en\_exprimido2<='1';centrifuga<='1'; bomba<='1';    when tm=> sd<=delicado;sn<=normal;st<=toalla;sj<=jeans;en\_enjuague2<='1';centrifuga<='0'; bomba<='0';    when tn=> sd<=delicado;sn<=normal;st<=toalla;sj<=jeans;en\_exprimidofinal<='1';centrifuga<='1'; bomba<='1';  when tp=> sd<=delicado;sn<=normal;st<=toalla;sj<=jeans;en\_desague<='1';bomba<='1';    end case;  end process;  end solucion;    **Descripción de VHDL**  El controlador o MSS del Sistema Digital está compuesto por diferentes entradas y salidas, las cuales se describen a continuación:  Entrada de **Reloj (Clock)** que es la característica principal de esta máquina secuencial, permite el cambio de estado conforme se detecten los flancos de reloj .  Señal de **RESETn** que permite regresar al primer estado en cualquier momento del programa, esta señal es asíncrona.  **Entradas STD\_LOGIC.**  **Encender:** esta entrada es el botón ON del sistema, permite inicializar toda la MSS pasando de su estado inicial a las diferentes acciones.  **Start**: Una vez escogidas todas las opciones del programa de lavado y diferentes tiempos, con el botón START podemos dar inicio al funcionamiento de la máquina, es decir empieza a trabajar según se haya programado.  *Entradas de selección de programa:* La MSS cuenta con un banco de switch que nos permite seleccionar los diferentes programas de lavados los cuales constan de diferentes tiempo y acciones según sea la prenda que se introduzca en la máquina.  Además, el controlador posee una entrada que permite adicionar un tiempo de remojo de la prenda, si el usuario así lo desea. Esta entrada se detecta mediante una botonera denominada Remojado.  Los diferentes programas de lavado con los que cuenta la MSS son:   * Jeans * Toalla * Normal * Delicado   Lleno: es la señal proveniente del sensor ubicado en el tanque de la lavadora que nos indica que el nivel de agua es el adecuado para empezar las operaciones de lavado.    *Entradas de Tiempos de Operación:* estas entradas no están conectadas a periféricos, como es el caso de botoneras o switch, pero son de gran importancia debido a que son estas entradas las que permitirán conocer que una tarea a culminado y se debe continuar con la siguiente según el programa escogido. Las entradas de tiempos de operación de la MSS son las siguientes, las cuales dependerán de los tiempos de programa seleccionado por el usuario para pasar de un estado bajo a un estado alto.   * t\_remojado * t\_lavado * t\_exprimido * t\_enjuague * t\_exprimido2 * t\_enjuague2 * t\_desague   **Salidas STD\_LOGIC.**  *Salidas Selectoras:* las denominamos selectoras debido a que estas ingresaran a diferentes MUXs con la finalidad de escoger entre los diferentes tiempos que posee cada programa de lavada que el usuario puede elegir para las distintas prendas a lavar.  Las señales selectoras que genera la MSS son:   * sr30 * sd * sn * st * sj   *Salidas Habilitadoras:* son señales generadas por el controlador con la finalidad de realizar alguna acción, es decir serán entradas habilitadoras de contadores de tal manera que podamos dar el tiempo justo para cada programa de lavado según lo que el usuario ha escogido inicialmente. Además, el controlador envía una señal habilitadora para iniciar la medición del nivel de agua en el tanque de tal manera que inicie el funcionamiento solo si se ha llegado a la referencia o nivel óptimo.  Las señales que genera el controlador son las siguientes:   * en\_nivel, * en\_remojado * en\_lavado * en\_exprimido * en\_enjuague * en\_exprimido2 * en\_enjuague2 * en\_desague   *Salidas de Acciones:* son las salidas destinadas al control o indicador de algún proceso, como el encendido de:   * válvula * centrifuga |

**Código VHDL completo de todos los bloques a utilizar en el Sistema Digital** (con su respectiva explicación a detalle).

|  |
| --- |
| **ANTIREBOTE** |
| LIBRARY IEEE;  USE IEEE.STD\_LOGIC\_1164.all;  USE IEEE.STD\_LOGIC\_ARITH.all;  USE IEEE.STD\_LOGIC\_UNSIGNED.all;  -- Debounce Pushbutton: Filters out mechanical switch bounce for around 40Ms.  ENTITY ANTIREBOTE IS  PORT(PB\_N, CLOCK\_100Hz : IN STD\_LOGIC;  PB\_SIN\_REBOTE : OUT STD\_LOGIC);  END ANTIREBOTE;  ARCHITECTURE a OF ANTIREBOTE IS  SIGNAL SHIFT\_PB : STD\_LOGIC\_VECTOR(3 DOWNTO 0);  BEGIN  -- Debounce clock should be approximately 10ms or 100Hz  PROCESS  BEGIN  WAIT UNTIL (clock\_100Hz'EVENT) AND (clock\_100Hz = '1');  -- Use a shift register to filter switch contact bounce  SHIFT\_PB(2 DOWNTO 0) <= SHIFT\_PB(3 DOWNTO 1);  SHIFT\_PB(3) <= NOT PB\_N;  IF SHIFT\_PB(3 DOWNTO 0)="0000" THEN  PB\_SIN\_REBOTE <= '0';  ELSE  PB\_SIN\_REBOTE <= '1';  END IF;  END PROCESS; | | |

|  |
| --- |
| **Descripción:** los pulsadores tardan cierto tiempo en alcanzar un estado estable, es decir, que durante muy poco tiempo (del orden de los milisengundos) pueden cambiar entre el estado pulsado y no pulsado hasta que la señal se estabiliza. A pesar de ser muy poco tiempo, es el suficiente para que un sistema digital lo detecte como varias pulsaciones. El bloque anti rebote permite validar que los pulsadores solo han sido presionados una vez y no varias. |

|  |
| --- |
| **DIVISOR DE FRECUENCIA** |

|  |
| --- |
| LIBRARY IEEE;  USE IEEE.STD\_LOGIC\_1164.all;  USE IEEE.STD\_LOGIC\_ARITH.all;  USE IEEE.STD\_LOGIC\_UNSIGNED.all;  ENTITY CLOCK\_DIV\_50 IS  PORT  ( CLOCK\_50MHz :IN STD\_LOGIC;  CLOCK\_1MHz :OUT STD\_LOGIC;  CLOCK\_100KHz :OUT STD\_LOGIC;  CLOCK\_10KHz :OUT STD\_LOGIC;  CLOCK\_1KHz :OUT STD\_LOGIC;  CLOCK\_100Hz :OUT STD\_LOGIC;  CLOCK\_10Hz :OUT STD\_LOGIC;  CLOCK\_1Hz :OUT STD\_LOGIC);  END CLOCK\_DIV\_50;  ARCHITECTURE a OF CLOCK\_DIV\_50 IS  SIGNAL count\_1Mhz: STD\_LOGIC\_VECTOR(5 DOWNTO 0);  SIGNAL count\_100Khz, count\_10Khz, count\_1Khz: STD\_LOGIC\_VECTOR(2 DOWNTO 0);  SIGNAL count\_100hz, count\_10hz, count\_1hz: STD\_LOGIC\_VECTOR(2 DOWNTO 0);  SIGNAL clock\_1Mhz\_int, clock\_100Khz\_int, clock\_10Khz\_int, clock\_1Khz\_int: STD\_LOGIC;  SIGNAL clock\_100hz\_int, clock\_10hz\_int, clock\_1hz\_int: STD\_LOGIC;  BEGIN  PROCESS  BEGIN  -- Divide by 50  WAIT UNTIL clock\_50Mhz'EVENT and clock\_50Mhz = '1'; -- 24 Mhz  IF count\_1Mhz < 49 THEN  count\_1Mhz <= count\_1Mhz + 1;  ELSE  count\_1Mhz <= "000000";  END IF;  IF count\_1Mhz < 4 THEN  clock\_1Mhz\_int <= '0';  ELSE  clock\_1Mhz\_int <= '1';  END IF;  -- Ripple clocks are used in this code to save prescalar hardware  -- Sync all clock prescalar outputs back to master clock signal  clock\_1Mhz <= clock\_1Mhz\_int;  clock\_100Khz <= clock\_100Khz\_int;  clock\_10Khz <= clock\_10Khz\_int;  clock\_1Khz <= clock\_1Khz\_int;  clock\_100hz <= clock\_100hz\_int;  clock\_10hz <= clock\_10hz\_int;  clock\_1hz <= clock\_1hz\_int;  END PROCESS;  -- Divide by 10  PROCESS  BEGIN  WAIT UNTIL clock\_1Mhz\_int'EVENT and clock\_1Mhz\_int = '1';  IF count\_100Khz /= 4 THEN  count\_100Khz <= count\_100Khz + 1;  ELSE  count\_100Khz <= "000";  clock\_100Khz\_int <= NOT clock\_100Khz\_int;  END IF;  END PROCESS;  -- Divide by 10  PROCESS  BEGIN  WAIT UNTIL clock\_100Khz\_int'EVENT and clock\_100Khz\_int = '1';  IF count\_10Khz /= 4 THEN  count\_10Khz <= count\_10Khz + 1;  ELSE  count\_10Khz <= "000";  clock\_10Khz\_int <= NOT clock\_10Khz\_int;  END IF;  END PROCESS;  -- Divide by 10  PROCESS  BEGIN  WAIT UNTIL clock\_10Khz\_int'EVENT and clock\_10Khz\_int = '1';  IF count\_1Khz /= 4 THEN  count\_1Khz <= count\_1Khz + 1;  ELSE  count\_1Khz <= "000";  clock\_1Khz\_int <= NOT clock\_1Khz\_int;  END IF;  END PROCESS;  -- Divide by 10  PROCESS  BEGIN  WAIT UNTIL clock\_1Khz\_int'EVENT and clock\_1Khz\_int = '1';  IF count\_100hz /= 4 THEN  count\_100hz <= count\_100hz + 1;  ELSE  count\_100hz <= "000";  clock\_100hz\_int <= NOT clock\_100hz\_int;  END IF;  END PROCESS;  -- Divide by 10  PROCESS  BEGIN  WAIT UNTIL clock\_100hz\_int'EVENT and clock\_100hz\_int = '1';  IF count\_10hz /= 4 THEN  count\_10hz <= count\_10hz + 1;  ELSE  count\_10hz <= "000";  clock\_10hz\_int <= NOT clock\_10hz\_int;  END IF;  END PROCESS;  -- Divide by 10  PROCESS  BEGIN  WAIT UNTIL clock\_10hz\_int'EVENT and clock\_10hz\_int = '1';  IF count\_1hz /= 4 THEN  count\_1hz <= count\_1hz + 1;  ELSE  count\_1hz <= "000";  clock\_1hz\_int <= NOT clock\_1hz\_int;  END IF;  END PROCESS;  END a; |

|  |
| --- |
| **Descripción:**  Inicialmente utilizamos el clock de la tarjeta DE0-nano que trabaja a 50MHz y a partir del divisor de frecuencia se obtiene un clock que trabaja a 1Hz de tal manera que al ejecutar el programa será posible observar las diferentes salidas que se tienen en los diferentes estados. |

|  |
| --- |
| **MUX – MULTIPLEXORES** |

|  |
| --- |
| library ieee;  use ieee.std\_logic\_1164.all;  entity Mux4a1 is  generic(n: integer:=4);  port( sel : in std\_logic\_vector(2 downTo 0);  ent0, ent1, ent2, ent3, ent4, ent5, ent6, ent7 : in std\_logic\_vector(n-1 downTo 0);  s : out std\_logic\_vector(n-1 downTo 0));  end Mux4a1;  Architecture sol of Mux4a1 is  Begin  with sel select  s <= ent0 when "000",  ent1 when "001",  ent2 when "010",  ent3 when "011",  ent4 when "100",  ent5 when "101",  ent6 when "110",  ent7 when "111";  end sol;  library ieee;  use ieee.std\_logic\_1164.all;  entity Mux2a1 is  generic(n: integer:=4);  port( sel : in std\_logic;  ent0, ent1 : in std\_logic\_vector(n-1 downTo 0);  s : out std\_logic\_vector(n-1 downTo 0));  end Mux2a1;  Architecture sol of Mux2a1 is  Begin  with sel select  s <= ent0 when '0',  ent1 when others;  end sol; |

|  |
| --- |
| **Descripción:**  Usamos de dos tipos:   * MUX 8 a 1 * MUX 2 a 1   Estos multiplexores nos permiten escoger entre diferentes valores de entradas mediante señales seleccionadoras las cuales irán aumentando según el numero de entradas a escoger en el MUX. Los valores que necesitamos son los diferentes tiempos de cada programa, estos serán escogido según el programa que desee el usuario. |

|  |
| --- |
| **COMPARADOR** |

|  |
| --- |
| library ieee;  use ieee.std\_logic\_1164.all;  Entity comparador is  Generic(n: integer:=4);  Port(A, B : in std\_logic\_vector(n-1 downTo 0);  men, igu, may : out std\_logic);  end comparador;  Architecture sol of comparador is  Begin  men <= '1' when A < B else '0';  igu <= '1' when A = B else '0';  may <= '1' when A > B else '0';  end sol; |

|  |
| --- |
| **Descripción**  Como indica su nombre nos permitirá conocer si dos valores, de varios bits, son iguales o no, además de conocer cual de ellos es mayor o menor según lo que se requiera.  Son de gran ayuda para conocer si se ha concluido el tiempo de programa comparando, siempre que este habilitado, el tiempo que está funcionando la maquina con el tiempo seteado por el usuario al inicio de la configuración. |

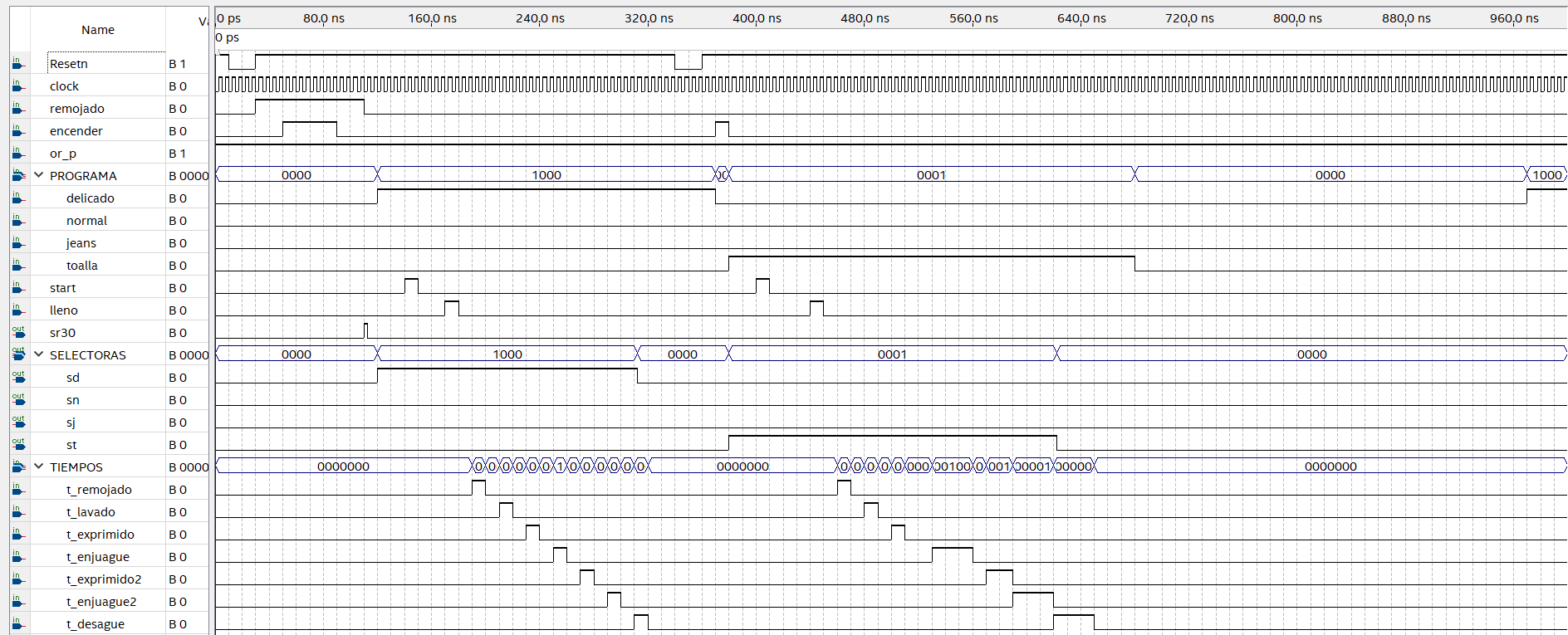
|  |
| --- |
| **CONTADOR UP** |

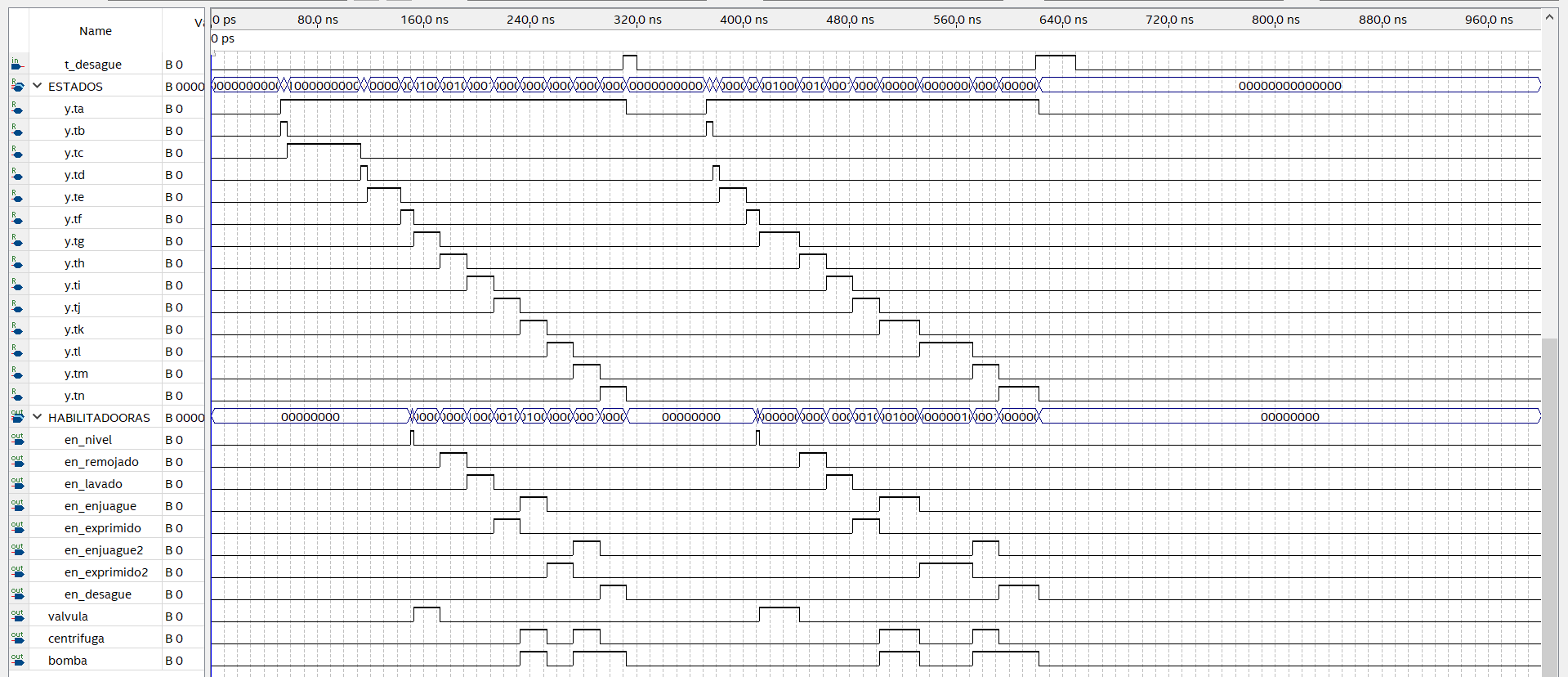
|  |
| --- |
| library ieee;  use ieee.std\_logic\_1164.all;  use ieee.std\_logic\_unsigned.all;  Entity contador\_up is  generic(n: integer:=4);  Port(Resetn, clock, En, Ld: in std\_logic;  Ent: in std\_logic\_vector(n-1 downTo 0);  Q: Buffer std\_logic\_vector(n-1 downTo 0));  End contador\_up;  Architecture sol of contador\_up is  Begin  Process(Resetn, clock)  Begin  if resetn = '0' then Q <= (others => '0');  elsif clock'event and clock = '1' then  if En = '1' then  if Ld = '1' then Q <= Ent;  else Q <= Q+1;  end if;  end if;  end if;  end Process;  end sol; |

|  |
| --- |
| **Descripción**  Como dice su nombre, se trata de un contador binario ascendente con el cual podemos llevar un registro del tiempo que esta funcionando la maquina y así controlar cada programa de lavado en conjunto con comparadores. |

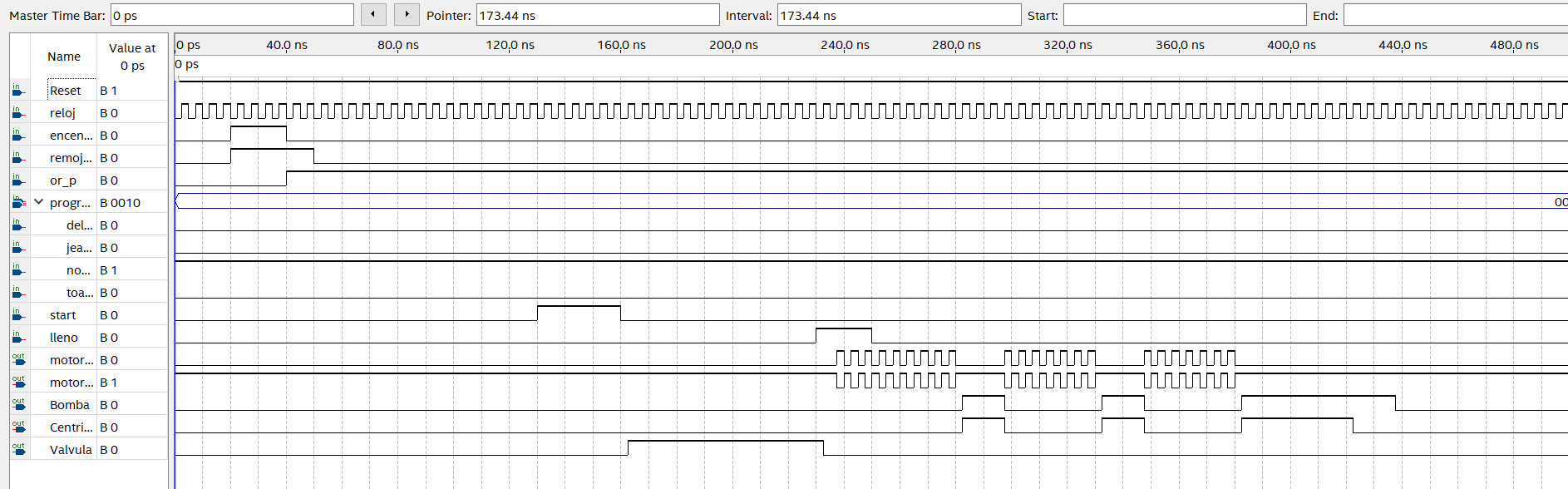
|  |
| --- |
| **ADC** |
| // adc.v  // Generated using ACDS version 17.0 595  `timescale 1 ps / 1 ps  module adc (  input wire CLOCK, // clk.clk  output wire ADC\_SCLK, // external\_interface.SCLK  output wire ADC\_CS\_N, // .CS\_N  input wire ADC\_DOUT, // .DOUT  output wire ADC\_DIN, // .DIN  output wire [11:0] CH0, // readings.CH0  output wire [11:0] CH1, // .CH1  output wire [11:0] CH2, // .CH2  output wire [11:0] CH3, // .CH3  output wire [11:0] CH4, // .CH4  output wire [11:0] CH5, // .CH5  output wire [11:0] CH6, // .CH6  output wire [11:0] CH7, // .CH7  input wire RESET // reset.reset  );  adc\_adc\_mega\_0 #(  .board ("DE0-Nano"),  .board\_rev ("Autodetect"),  .tsclk (16),  .numch (8),  .max10pllmultby (1),  .max10plldivby (1)  ) adc\_mega\_0 (  .CLOCK (CLOCK), // clk.clk  .RESET (RESET), // reset.reset  .CH0 (CH0), // readings.export  .CH1 (CH1), // .export  .CH2 (CH2), // .export  .CH3 (CH3), // .export  .CH4 (CH4), // .export  .CH5 (CH5), // .export  .CH6 (CH6), // .export  .CH7 (CH7), // .export  .ADC\_SCLK (ADC\_SCLK), // external\_interface.export  .ADC\_CS\_N (ADC\_CS\_N), // .export  .ADC\_DOUT (ADC\_DOUT), // .export  .ADC\_DIN (ADC\_DIN) // .export  );  endmodule |

|  |
| --- |
| **Descripción**  El convertidor Analógico-Digital nos permite traducir los datos obtenidos en tiempo real de la medición del nivel de agua en el tanque y llevarlo a un lenguaje binario reconocido por la programación de bloques y la MSS de tal manera que pueda activar o no la señal de LLENO. |

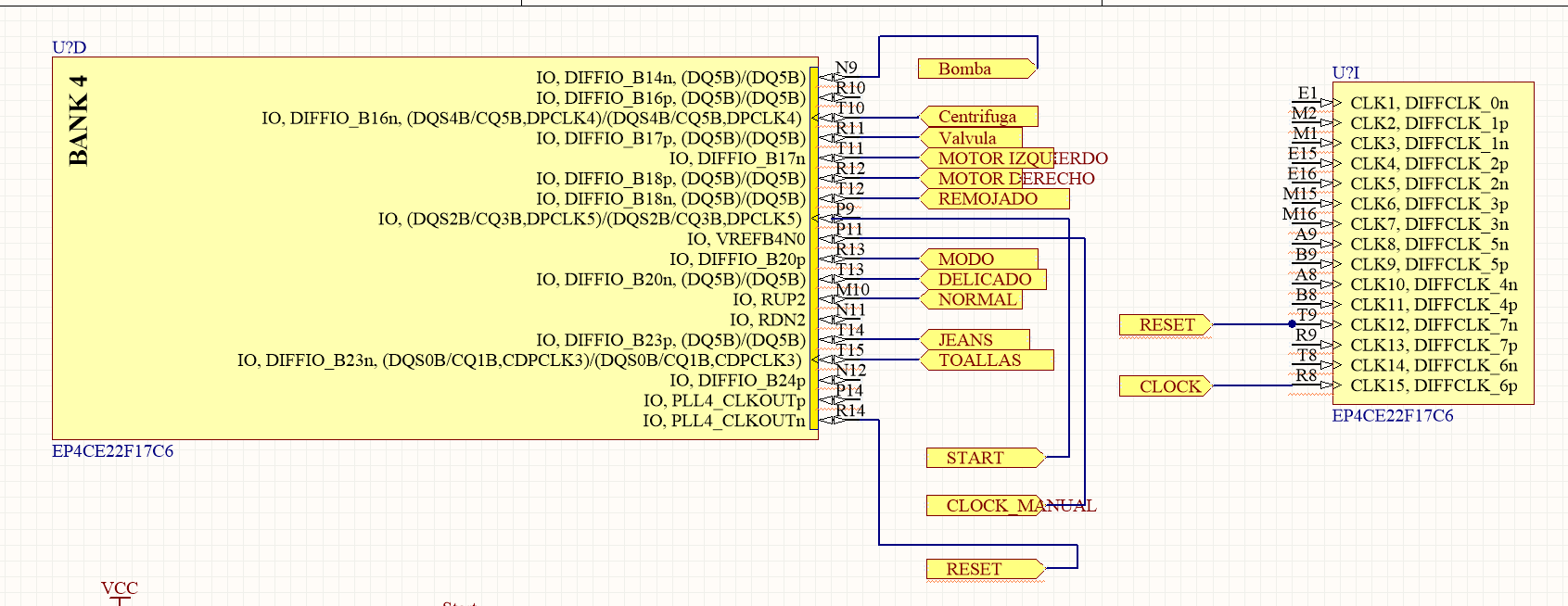
**Diagrama de tiempo del controlador** generado en el archivo .VWF (Vector Waveform File), en el que demuestre todos los estados de la MSS (con su respectiva explicación a detalle).

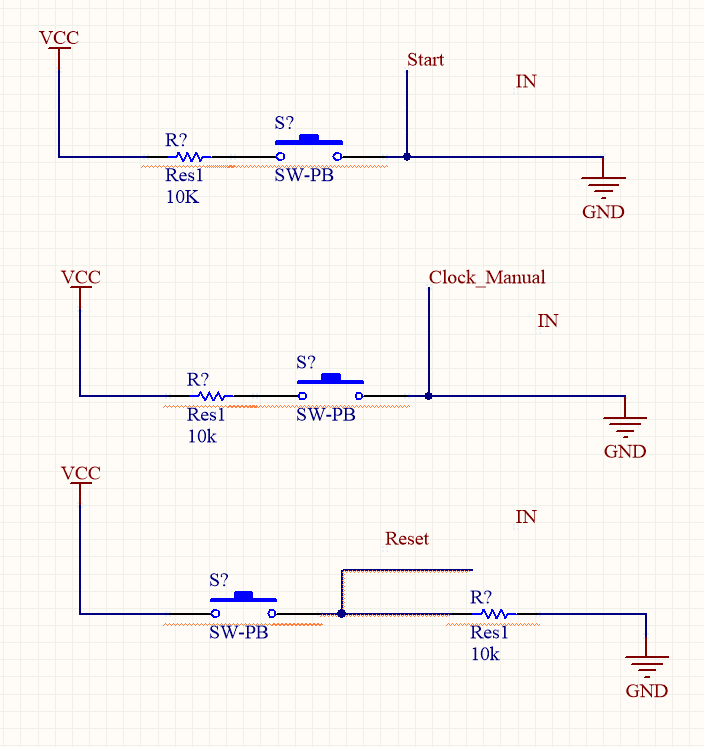
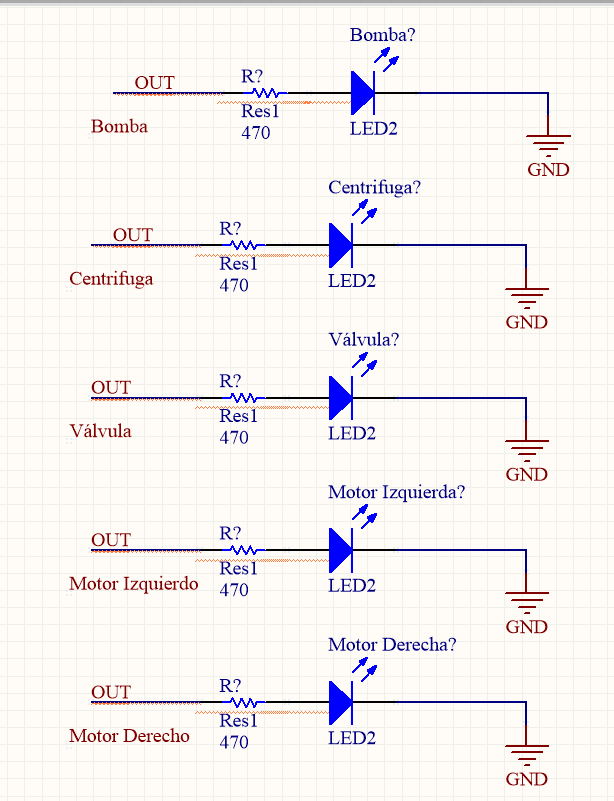


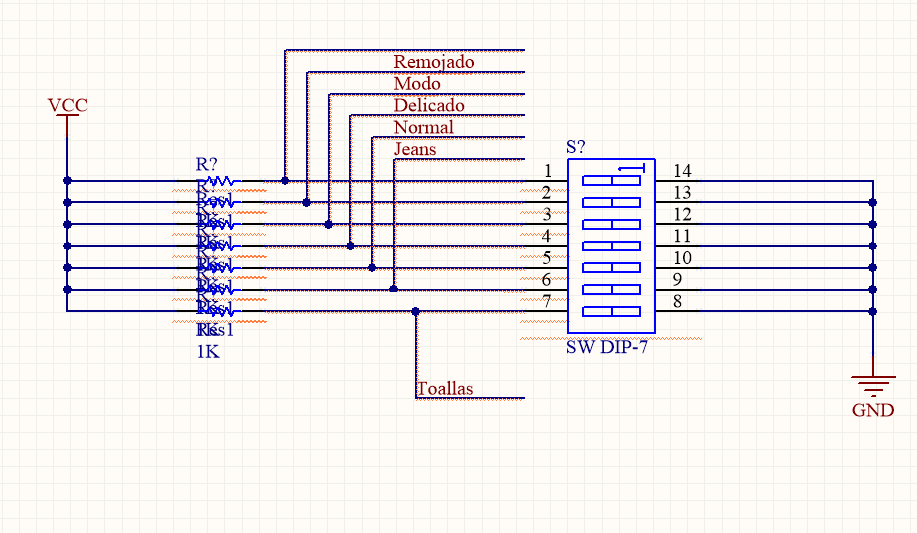
|  |
| --- |
| **Descripción**  Se inicia el programa con el botón de ‘encendido’ y se escoge la opción de remojo, esta opción de remojo agrega 30 min mas de lavado.  Luego se procede a escoger el tipo de programa de lavado a ejecutarse y al seleccionar cualquiera de estos se activará la señal ‘or\_p’, además de acuerdo con el programa de lavado escogido se encenderá una habilitadora de un mux donde se encuentran guardados los diferentes tiempos de lavado de acuerdo al tipo de programa.  Por otro lado, se tendrá de salida a ‘valvula’ la misma que se activa en el momento de llenado del tanque, además la salida ‘centrifuga’ que se activa en la etapa de exprimido y finalmente la salida ‘bomba’ que se activa en la etapa de exprimido y drenado. |

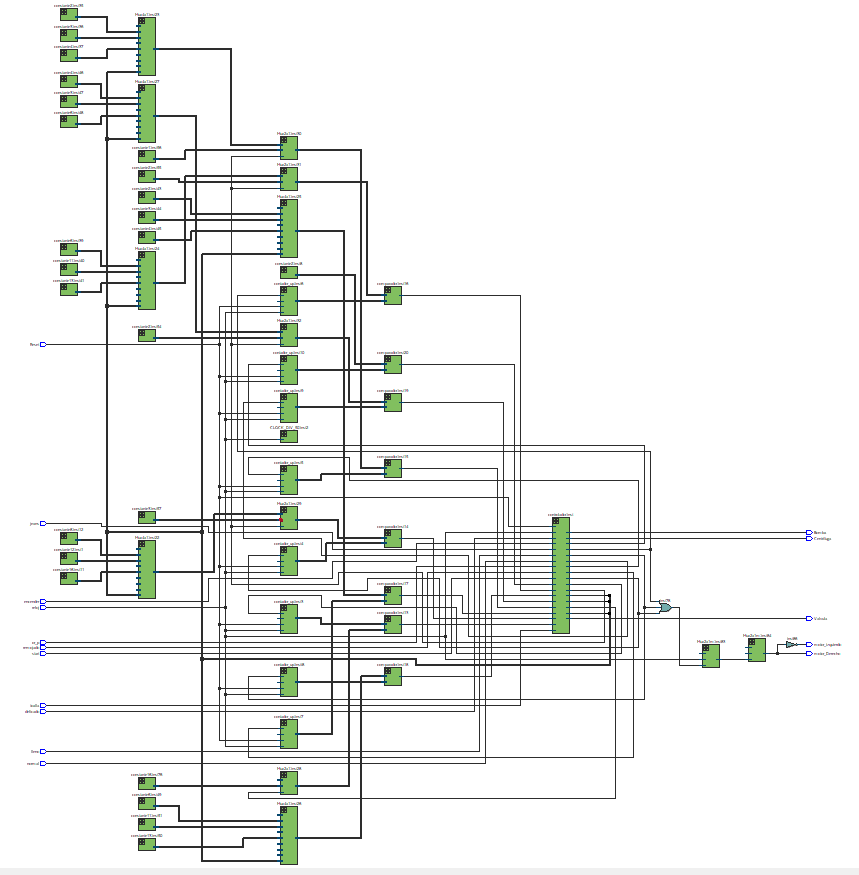
**Diagrama de tiempo del sistema digital** generado en el archivo .VWF (Vector Waveform File), en el que demuestre todos los instantes de su funcionamiento (con su respectiva explicación a detalle).

|  |
| --- |
| Descripción  Se inicia el programa con el botón de ‘encendido’ y se escoge la opción de remojo. Luego se debe de escoger algún tipo de lavado que se desea realizar, una vez escogido el programa se activará la señal ‘or\_p’ que da paso a la ejecución de los diferentes estados (esta señal esta descrita más detalladamente en el controlador).  A continuación, se presiona la botonera de ‘start’ de tal manera que se enciende la salida ‘válvula’ y una vez que se active la señal ‘lleno’ la salida ‘válvula’ se apaga.  Una vez lleno el tanque con agua se inicia el programa escogido con sus respectivos tiempos, teniendo en cuenta que para la etapa de lavado y enjuague se encenderán dos motores y para el drenado y exprimido se encenderá la salida de bomba y centrifuga. |

**Diagrama Esquemático** del proyecto en físico elaborado con la herramienta ALTIUM Designer (con su respectiva explicación a detalle).





**CONCLUSIONES**

* Utilizamos la herramienta Qsys del programa Quartus para la creación del bloque correspondiente al convertidor analógico digital de la tarjeta de desarrollo DE0-Nano.
* El sistema implementado en la tarjeta de desarrollo esta sincronizada al reloj de la tarjeta además de poseer un clock manual que permite visualizar de manera mas efectiva el cambio de estados del controlador diseñado.
* Debido a que no tenemos de enviar y recibir datos a grandes velocidades se coloco un bloque divisor de frecuencia en el que cambiamos la frecuencia de trabajo a 1 Hz.

**RECOMENDACIONES**

* Utilizar la herramienta Signal Tap del programa Quartus para observar el funcionamiento en tiempo real de cada una de las señales en caso de presentar errores de funcionamiento.
* En caso de utilizar botoncera es necesario colocar bloques antirebotes. El banco de switch usados en esta implementación no utiliza bloque de antirebotes.

Detalles del proyecto en:

Link de GitHub:

<https://github.com/marfmora1402/DigitalesII.git>

Link de videos:

Explicación Detallada del proyecto  
<https://www.youtube.com/watch?v=YRfDd2As8Ok>