

Universidad ORT – Uruguay

Facultad de Ingeniería

Obligatorio 2 (VHDL)

Electrónica Digital

Alumnos:

Sebastián Nicolás Effa Gallego (193248) – Ingeniería en Electrónica

Fernando Agustín Hernández Gobertti (173631) – Ingeniería en Telecomunicaciones

Docentes:

Diego Sáez Schwedt

Gonzalo Óscar Costa Gnocchi

Pablo Sebastián Miller Balian

Fecha: 24 de noviembre del 2015

Índice del Informe

Objetivo del Obligatorio 2:	4
Diagrama Global de los Componentes Programados	6
Componentes Programados	6
Explicativa General.....	9
Decodificador y Display de 7 Segmentos.....	11
Transmisión de Comunicación Serial	14
RS 232	14
Registro de Desplazamiento (<i>Shift-Register</i>)	17
Máquina de Estados.....	19
Primer, Segundo y Tercer Byte Recibido: Direccionamiento, Comando y Datos.....	21
Codificación de los Programas Solicitados	25
Contadores Concatenados.....	25
Programa 1	27
Programa 2	31
Programa 3	34
Programa 4	37
Programa 5	40
Programa 6	43
Programa 7	46
Programa 8	49
Compilación de la Lógica de los Programas	52

Despliegue de Pantallas Requeridas	56
Convertidor de Bits a BCD.....	56
Pantalla 1	57
Pantalla 2	58
Pantalla 3	60
Pantalla 4	61
 Moduladores de la Señal XTAL de 50 MHz de la placa DE0 FPGA	65
Concepto de Modulador o Divisor de Frecuencia.....	65
Modulador de 10 Hz.....	68
Modulador de 0.5 Hz.....	69
Modulador de 2 Hz	70
 Macro Bloques.....	71
Macro Programa.....	71
Macro Pantalla.....	74
Mezcladora.....	76
Obligatorio 2 (<i>Top-Level Entity</i>)	80
 Especificaciones Finales	87
Multiplicador Binario	87
Animaciones en los <i>Displays</i> de 7 Segmentos.....	88
Botones Sin Retención	90
 Tabla de Prueba	91
 Diagrama de Tiempos	97
 Referencias Bibliográficas	99

1. Objetivo del Obligatorio 2:

Se desea la implementación en una placa FPGA del tipo *Cyclone III* de Altera en Lenguaje de Descripción de *Hardware* (VHDL), un programa consistente en la muestra de actividad de cuatro mezcladoras independientes o consideradas de a conjuntos (grupos) controlados por el usuario. El entorno de programación trabajado está brindado por el software de nombre “*Quartus II Web Edition versión 9.0 SP2*” ofrecido por los dispositivos de la facultad (descargado de: https://wl.altera.com/download/quartus-ii-we/dnl-quartus_we-v90.jsp).

Por el programa general implementado se las debió poder llenar con líquido expulsado por cuatro válvulas independientes, aplicar presión (compresión de la sustancia), mezclar el contenido del recipiente (en sentido horario y antihorario), calentar la mezcla por una resistencia colocada en el interior del receptor y finalmente possibilitar el vaciado de la misma. Lógicamente, estas acciones solo serán desplegadas en los *displays* (pantallas de siete segmentos) y LED's (Diodos Emisores de Luz) de la FPGA, sin una visualización en base a acciones en mezcladoras tangibles.

El protocolo a seguir correspondiente al orden y forma en que las acciones anteriormente referidas se prosiguen es implementado por programas fijos y predefinidos estructurados por los docentes y descritos en la hoja del obligatorio correspondiente. Estos ocho programas a implementar en la placa FPGA debieron funcionar en forma también independiente y en el momento indicado por el usuario, junto con un correcto y coherente despliegue de información en los LED's y *displays* de 7 segmentos según cierta estructura también explicitada en la hoja explicativa del trabajo. De esta manera, se tuvo que implementar 4 formatos de pantallas básicas en las que se muestren los datos arribados por el programa activado junto a otra que despliegue información sobre un estado general de las cuatro mezcladoras y la aparición de un mensaje de error en caso de ocurrir un problema (botón de emergencia) o si la temperatura de la mezcla (valor de sensor) es superior a aquella previamente definida con la que funcionaría el sistema.

Dentro de los formatos de pantallas mencionados se debió mostrar el tiempo transcurrido en décimas, segundos y decenas de segundos, junto a una animación vinculada al estado del programa en el que se encuentre (llenado, compresión, calentamiento o mezcla), junto al despliegue de información referente al porcentaje de cuanto llene cada válvula y el nivel de temperatura y velocidad actual de la

mezcla. Todas las acciones realizadas debieron estar relacionadas de forma coherente y previamente descritos a los *switchs* y pulsadores accionados.

Los componentes de la placa FPGA utilizados explícitamente (sin referirse a los componentes internos del dispositivo) para visualizar lo solicitado, correspondiente a las acciones programadas que se precisa que realice cada mezcladora, fueron:

- 10 LED's de color verde
 - **Pines:** J1, J2, J3, H1, F2, E1, C1, C2, B2, B1.
- 3 pulsadores o botones
 - **Pines:** H2, G3, F1.
- 10 *switchs*
 - **Pines:** J6, H5, H6, G4, G5, J7, H7, E3, E4, D2.
- Entrada del Reloj (XTAL 50 MHz)
 - **Pines:** G21.
- 4 *displays* o pantallas de 7 segmentos (HEX3, HEX2, HEX1, HEX0)
 - **Pines (HEX3):** B18, F15, A19, B19, C19, D19, G15, G16.
 - **Pines (HEX2):** D15, A16, B16, E15, A17, B17, F14, A18.
 - **Pines (HEX1):** A13, B13, C13, A14, B14, E14, A15, B15.
 - **Pines (HEX0):** E11, F11, H12, H13, G12, F12, F13, D13.

2. Diagrama Global de los Componentes Programados

2.1. Componentes Programados

A fin de realizar el trabajo asignado se prosiguió a crear una serie de herramientas, programadas y nombrados como bloques durante toda la extensión del documento, las cuales eran partícipes cada una en su medida y con su indicada responsabilidad en el correcto funcionamiento de la experiencia. Algunos de estos componentes, totalizando aproximadamente 68 archivos correspondientes a la misma cantidad de bloques (incluyendo marco bloques), tales como ciertos multiplexores y moduladores (sirviendo para fomentar el modelo básico de programar un componente de tal índole) fueron programados de la forma indicada en el documento brindado por los docentes y descargado del sitio *moodle* de Aulas de la facultad de nombre Laboratorio 1 de VHDL, especialmente en aquellas dos secciones bajo el título “Introducción a VHDL” y “Sistema 01”, comprendida entre las páginas 6 a 13, y 14 a 17 respectivamente. Los archivos creados, a explicar su funcionalidad en futuras puntuaciones del actual documento, fueron los desplegados a continuación:

- Lectura de Bytes:
 - shift_register_1byte
 - shift_register_3bytes
 - serial_232
 - direccionar
 - comandar

- Animaciones:
 - animación_llenado
 - animación_compresion
 - animación_giro_antihorario
 - animación_giro_horario

- Comparadores:
 - comparador_2bits
 - comparador_4bits
 - comparador_8bits
 - comparador_16bits
 - compararCuenta

- Convertidor bits a BCD:
 - agregar_4bits
 - bits_to_BCD

- Decodificadores:
 - deco2a4
 - deco3a8
 - deco4a16
 - deco7segmentos

- Multiplexores:
 - mux2a1_4bits
 - mux2a1_8bits
 - mux2a1_16bits
 - mux4a1_1bit
 - mux4a1_4bits
 - mux4a1_8bits
 - mux8a1_1bit
 - mux8a1_4bits
 - mux16a1_1bit
 - mux16a1_2bits
 - mux16a1_4bits
 - mux16a1_8bits
 - mux32a1_1bit

- Flip Flop:
 - flip_flop_D_1bit
 - flip_flop_D_2bits
 - flip_flop_D_4bits
 - flip_flop_D_8bits
 - flip_flop_D_8bits_descendente
 - flip_flop_D_16bits

- Sumadores:
 - sumador_2bits
 - sumador_4bits
 - sumador_8bits
 - sumador_16bits

- Multiplicador:
 - Multiplicador_4bits

- Restadores:
 - restador_2bits
 - restador_4bits

- Contadores
 - contador_4bits
 - contador_8bits
 - contador_16bits
 - contador_decimal

- Moduladores:
 - modul
 - modul_05Hz
 - modul_2Hz
 - modul_10Hz

- Programas:
 - programa1
 - programa2
 - programa3
 - programa4
 - programa5
 - programa6
 - programa7
 - programa8
- Pantallas:
 - pantalla1
 - pantalla2
 - pantalla3
 - pantalla4
- Macro Bloques:
 - macroprograma
 - macropantalla
 - mezcladora
 - obligatorio2

2.2. Explicativa General

El procedimiento realizado a fin de mantener un programa prolífero y organizado fue el de realizar los programas VHDL en forma dividida y por partes, comenzando desde lo más sencillo e independiente hacia lo más complejo e interconectado (por ejemplo, los macro bloques). De todas formas, se debió imaginar y predisponer desde un comienzo cual sería la organización general del trabajo a realizar de tal modo de facilitar la consiguiente relación de datos correspondientes a las entradas y salidas de cada bloque, en estrecha correspondencia con la función primordial del mismo para con el programa global.

Así, partiendo desde la programación de la totalidad de las variantes de los componentes imaginados a utilizar en el futuro próximo, se comenzó programando los multiplexores, sumadores, restadores, comparadores, decodificadores y flip flop, cuya estructura de programación lógica básica se encontraba desde un comienzo disponible en el documento brindado por los docentes. Se eligieron estos componentes a su vez por la gran independencia en su funcionamiento respecto a lo que ocurre en el resto de los programas, buscando asemejar su funcionamiento al de los componentes trabajos a lo largo del curso y aplicados en demás laboratorios y obligatorios de forma física, conectándolos haciendo uso de la placa y soldando cada cable de forma individual. Se aprovecha la oportunidad para explicitar una ventaja clara y obvia de la programación lógica realizada, la cual consiste en el brindar la posibilidad de efectuar las conexiones sin la necesidad de soldar, acción que en el caso de necesitarla transformaría el trabajo en algo humanamente imposible de satisfacer.

A continuación, una vez probado el correcto funcionamiento de los mismos, se prosiguió a implementar los contadores y demás programas auxiliares de forma de ir aumentando gradualmente la complejidad de los programas realizados, tanto funcional como estructural. Por ejemplo, los contadores se programaron en base a instancias de multiplexores y flip flop, como posteriormente se analizara en detalle. Al referirse a los programas realizados de forma auxiliar se hace mención a aquellos tales como “modul”, “compararCuenta” y “agregar_4bits” los cuales poseen funcionamiento únicamente dependiente de otros programas, más primordiales y específicos que estos últimos, como método de organizar prolijamente el código entregado adjunto.

Otro de los programas en los cuales no se necesitó un funcionamiento previo de demás programas aparte de los básicos anteriores consistió en las animaciones, los cuales a base de un

registro de desplazamiento (creado mediante instancias de los flip flop antes programados y testeados) demostró funcionar apropiadamente de acuerdo a lo solicitado.

Continuando, luego de un exhaustivo estudio frente a la lectura de bytes y demás aplicaciones necesarias para proseguir, se buscó la creación de los programas de una complejidad interna, necesitando capacidades de deducción superior a los conocimientos de los estudiantes que escriben. De esta forma, al informarse de cómo se debían realizar cada acción, se continuó a programar el “multiplicador” necesario junto al “convertidor bits a BCD” y la totalidad de los programas VHDL encargados de decodificar, interpretar y desplegar los bytes recibidos por la terminal de transmisión serial RS 232, debiendo utilizar para estos últimos un osciloscopio digital en más de una oportunidad debido a problemas en la sincronización del *clock* con la cuenta necesitada para que el software determine el comienzo y final del byte a leer.

Una vez funcional lo planteado anteriormente, se prosiguió a la realización de los correspondientes ocho programas y cuatro pantallas necesarias para el despliegue de información, ya detectando el requerimiento de máquinas de estado para los programas mencionados y conociendo la debida integridad lógica que debían mantener los componentes entre sí. Aún así, continuando con el protocolo planteado al comienzo del punto, se testeó cada programa y pantalla de forma individual, brindando valores a modo de casos de prueba que permitan visualizar el correcto labor de los mismos antes de continuar con el trabajo en cuestión.

Finalmente, se programaron el macroprograma, la macropantalla, la mezcladora y el obligatorio2 (en esa disposición, pues consisten en un orden creciente de complejidad y jerarquización implementada para los bloques en cuanto a las responsabilidades de cada cual en el trabajo), considerados macro bloques al poseer varias instancias de cada bloque anteriormente realizado. Aquellas son, por tanto, las que posibilitan el adecuado y coherente despliegue de información, ligando las salidas de los programas con las entradas de las pantallas, mientras que indican la dependencia con los switchs y botones de la placa FPGA, entradas generales de todo el obligatorio junto a las salidas del mismo desplegadas en los LEDs y *displays* de la mencionada placa. De la misma forma, el obligatorio2, considerado como la *Top-Level Entity* del trabajo, fue la encargada de interrelacionar las acciones a realizar indicadas por el usuario a la vez que interconectar las mezcladoras y decidir la muestra de información en la FPGA.

3. Decodificador y Display de 7 Segmentos

Basándose en el decodificador de un display de 7 segmentos programado según lo descrito en el Laboratorio 1 de VHDL bajo el título de “Sistema 01” comprendidas entre las páginas 14, 15 y 16 del documento, se prosiguió a indicar en ciertas partes del programa que código mostrar en los mismos, siendo los posibles a desplegar con este método los referidos a números (al ingresar al decodificador vectores en binario de 4 bits y obtener en la salida señales de 8 bits prontas para ser conectadas a las terminales de los displays deseados correspondientes).

Sin embargo, para el obligatorio también se debió indicar a los displays el despliegue correcto de letras o demás símbolos como condiciones de letra, por lo que se tuvo que crear señales de 8 bits configuradas manualmente de forma adecuada según como las interpretarían las terminales del mismo una vez ejecutado el programa. De esta manera, considerando la numeración de los *displays* de la placa según la imagen a continuación y relacionándolas en un vector regido de la forma $[(DP) G F E D C B A] = [7 \ 6 \ 5 \ 4 \ 3 \ 2 \ 1 \ 0]$, nombrando al vector de esta manera de la forma “(7 downto 0)” y realizando el ingreso manual de la manera mostrada en la próxima tabla.

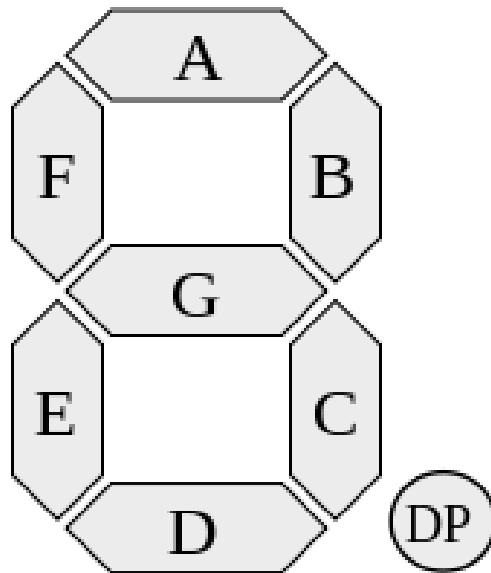
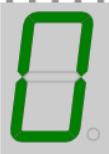
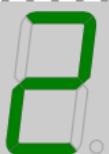


Figura 1. Normalización del indicador de pines utilizado en el trabajo y en la descripción del mismo a lo largo del documento

Considérese que por configuración predeterminada de los *displays* se encontró con que en lugar de ser la misma aquella que de forma intuitiva e implícita se considera como valor lógico 0 relacionado con LED apagado y 1 con LED prendido, los *displays* de la placa FPGA llevan un comportamiento opuesto, de tal modo que el valor lógico 0 significa LED prendido y valor lógico 1 con LED apagado. Esta configuración fue la mostrada en la tabla siguiente:

Letra o Símbolo	$[(DP) G F E D C B A] = [7 \ 6 \ 5 \ 4 \ 3 \ 2 \ 1 \ 0]$								Imagen demostrativa
Número 0	1	1	0	0	0	0	0	0	
Número 1	1	1	1	1	1	0	0	1	
Número 2	1	0	1	0	0	1	0	0	
Número 3	1	0	1	1	0	0	0	0	
Número 4	1	0	0	1	1	0	0	1	
Número 5	1	0	0	1	0	0	1	0	

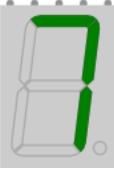
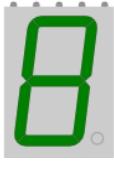
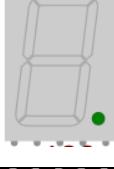
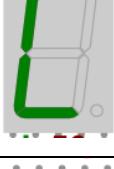
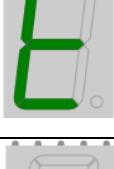
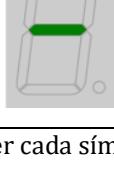
Número 6	1	0	0	0	0	0	1	0	
Número 7	1	1	1	1	1	0	0	0	
Número 8	1	0	0	0	0	0	0	0	
Número 9	1	0	0	1	1	0	0	0	
Punto Decimal (".")	0	1	1	1	1	1	1	1	
Letra "L"	1	1	0	0	0	1	1	1	
Letra "P"	1	0	0	0	1	1	0	0	
Letra "E"	1	0	0	0	0	1	1	0	
Símbolo “-“	1	0	1	1	1	1	1	1	

Tabla 1. Indicación de los valores a asignar al *display* de 7 segmentos de forma de obtener cada símbolo

4. Transmisión de Comunicación Serial

4.1. RS 232

La transmisión serial se realiza por el protocolo RS-232 (*Recommended Standard 232*) que designa una norma para el intercambio de datos binarios entre equipos. Consiste en un conector tipo DB-25 o el más extendido DE-9 para ciertos periféricos.

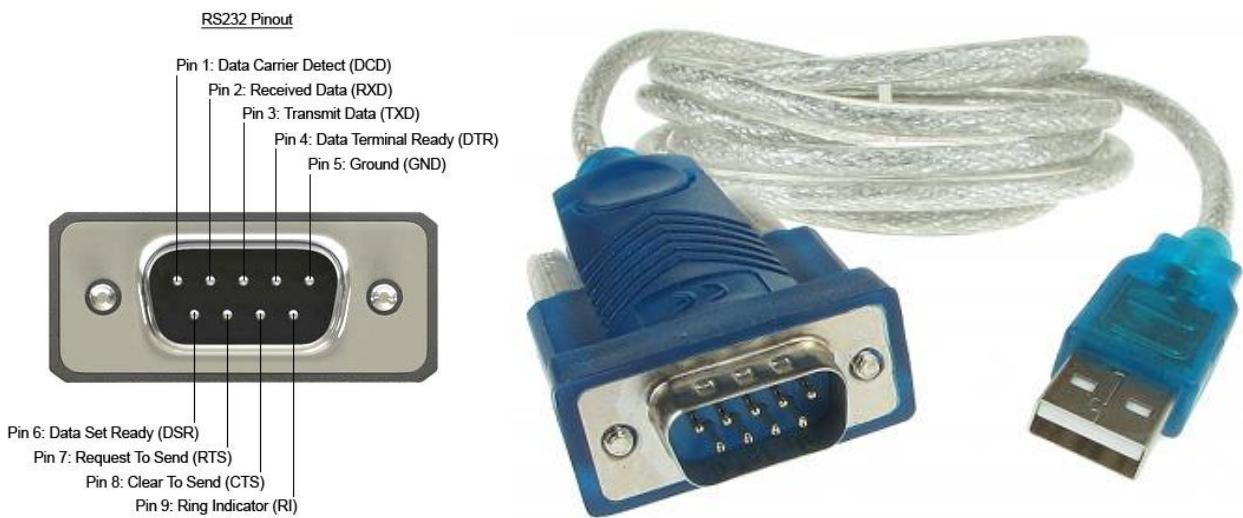


Figura 2. Muestra de los conectores usados en el trabajo para hacer posible la transmisión serial RS 232

La interfaz puede trabajar en comunicación asíncrona o síncrona y tipos de canal *simplex*, *half duplex* o *full duplex*. En un canal simplex los datos siempre viajarán en una dirección, por ejemplo desde DCE a DTE. En un canal *half duplex*, los datos pueden viajar en una u otra dirección, pero sólo durante un determinado periodo de tiempo; luego la línea debe ser conmutada antes que los datos puedan viajar en la otra dirección. En un canal *full duplex*, los datos pueden viajar en ambos sentidos simultáneamente. Las líneas de *handshaking* de la RS-232 se usan para resolver los problemas asociados con este modo de operación, tal como en qué dirección los datos deben viajar en un instante determinado.

Si un dispositivo de los que están conectados a una interfaz RS-232 procesa los datos a una velocidad menor de la que los recibe deben de conectarse las líneas *handshaking* que permiten realizar un control de flujo tal que al dispositivo más lento le de tiempo de procesar la información. Las líneas

de *handshaking* que permiten hacer este control de flujo son las líneas RTS y CTS. Los diseñadores del estándar no concibieron estas líneas para que funcionen de este modo, pero dada su utilidad en cada interfaz posterior se incluye este modo de uso.

En este caso solo se necesita enviar información desde un PC a las 4 mezcladoras (Placa Altera DE0) por lo que es suficiente utilizar un solo canal para el envío de datos en modo de transmisión asíncrona (con un terminal de tierra de referencia).

La línea de transmisión (Rx) se encuentra por defecto (reposo) en estado alto, al comenzar el envío de un byte (8 bits), ésta manda un bit de comienzo (*start bit*) con el cual se sincronizan los relojes del emisor y receptor y luego se prosigue a enviar los 8 bits.

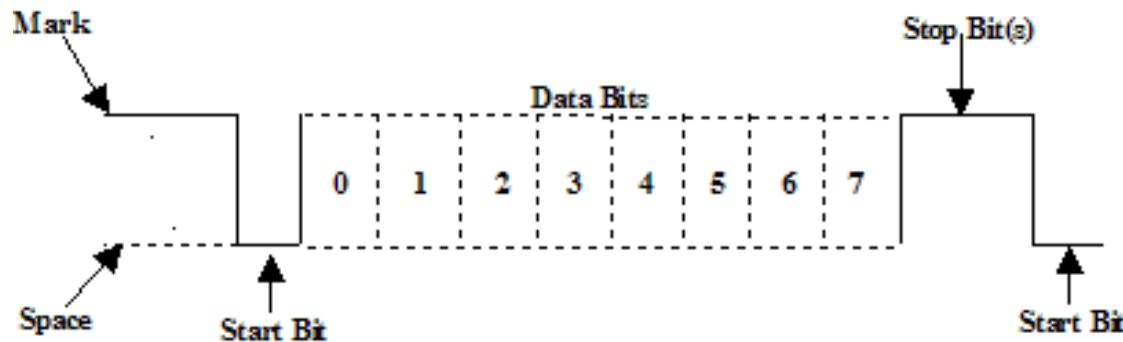


Figura 3. Forma de las señales enviadas a través del serial como es explicado en el documento

RS232 Transmission of the letter 'J'

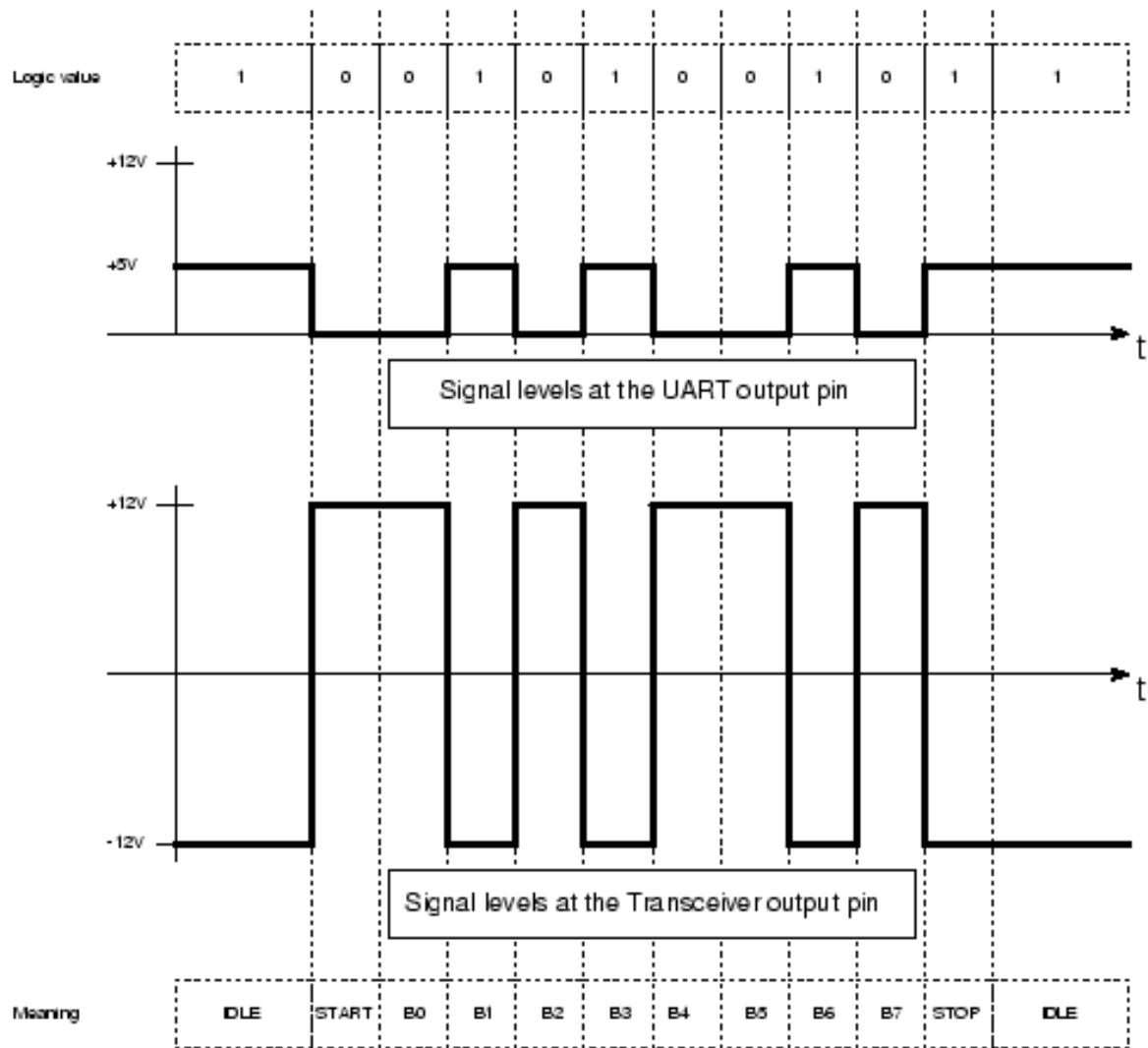


Figura 4. Señales en conjunto de la transmisión serial RS 232, de forma más explícita y desarrollada de manera exhaustiva.

4.2. Registro de Desplazamiento (*Shift-Register*)

Se conoce como registro de un circuito al ser capaz de almacenar un dato. El registro puede ser de 8 bits, 16 bits, 32 bits o los bits que se precisen para cada caso puntual. Se dicen de desplazamiento a los registros en los que el dato fluye de un bit a el siguiente en el proceso de la escritura o la lectura. Cada bit del registro se compone de un flip flop. La entrada de los datos en el registro puede ser en serie o en paralelo.

De acuerdo a lo buscado, se decidió mediante una máquina de estados, detectar el bit de *start* y activar un reloj que de acuerdo a la velocidad de la transmisión serial elegida, se encargará de sincronizar un contador conectado al reloj generado por la placa y con él detectar los 8 bits que se irán enviando una registradora que los almacenará. Luego de enviar un byte, éste se almacenará en otra registradora de bytes, cuando se obtienen los 3 bytes necesarios para las tareas a realizar, se prosigue con el funcionamiento del controlador de las mezcladoras.

El funcionamiento de las registradoras consiste en conectar flip flops de tipo D en cascada, es decir, conectar la salida del primer flip flop a la entrada del segundo y así consecutivamente según la cantidad de flip flops (y por tanto datos) que se requieran almacenar; siendo la entrada del primer flip flop la señal de entrada de la registradora y las conexiones internas junto con la salida del último flip flop, las señales de salida. Por último, todos los relojes de los flip flops se conectan en paralelo, éste, mediante un flanco ascendente, es decir, el pase de estado bajo al estado alto, se encargará de correr la señal de entrada de flip flop en flip flop. Una vez registrados los datos se congela el reloj de entrada para evitar tomar datos erróneos.

Según el dato que se quiera almacenar (ya sean bits, bytes o demás); se necesitarán flip flops de tantos bits como dimensión del dato a almacenar. Es decir, por ejemplo, para el caso de la registradora de bytes se usaron flip flops de 8 bits.

Ejemplo: *Shift Register* de 4 bits

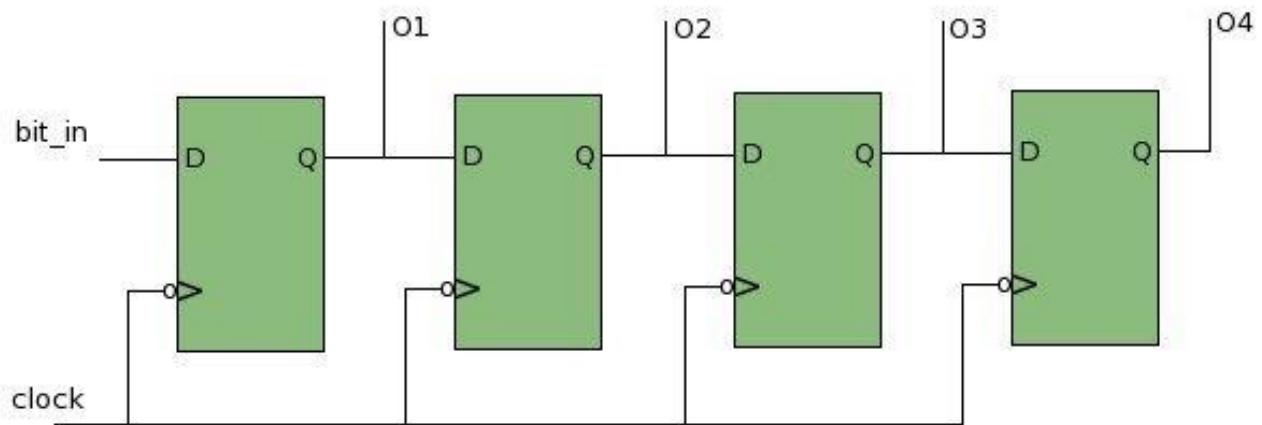


Figura 5. Diagrama de bloques de un registro de desplazamiento arbitrario, a modo de ejemplo del análogo realizado

4.3. Máquina de Estados

Con la claridad del sistema buscado, se prosiguió a realizar una máquina de estados para el proceso de captura y almacenamiento de los bytes.

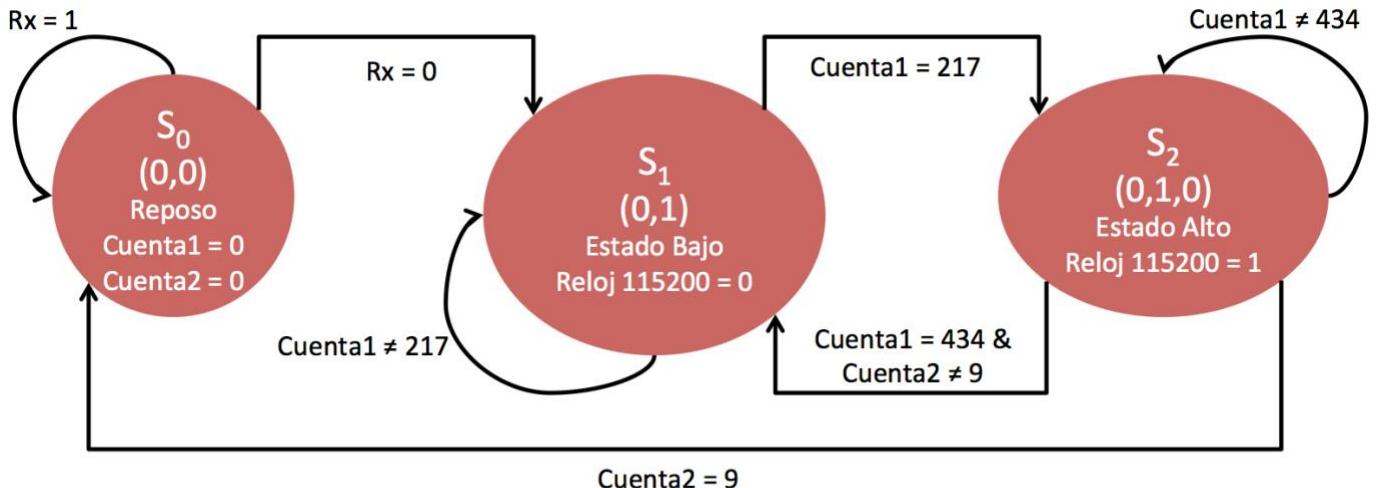


Figura 6. Diagrama de la Máquina de Estado utilizada para tratar la lectura de bytes mediante el uso de la transmisión de comunicación serial RS 232

Una vez que se tienen los 3 bytes se prosigue al procesamiento de éstos de acuerdo a la responsabilidad de cada uno.

Estado Actual			Entradas de los Flip-Flop tipo D		Condición			Estado Siguiente		
S_i	$Q_1(i)$	$Q_0(i)$	D_1	D_0	Rx	$Cuenta1$	$Cuenta2$	S_{i+1}	$Q_1(i+1)$	$Q_0(i+1)$
S_0	0	0	0	0	1	X	X	S_0	0	0
S_0	0	0	0	1	0	X	X	S_1	0	1
S_1	0	1	0	1	X	$Cuenta1 \neq 217$	X	S_1	0	1
S_1	0	1	1	0	X	$Cuenta1 = 217$	X	S_2	1	0
S_2	1	0	1	0	X	$Cuenta1 \neq 434$	X	S_2	1	0
S_2	1	0	1	1	X	X	$Cuenta2 = 9$	S_0	0	0
S_2	1	0	0	1	X	$Cuenta1 = 434$	$Cuenta2 \neq 9$	S_1	0	1
S_3	1	1	0	0	X	X	X	S_0	0	0

Tabla 2. Tabla de Verdad o Lógica referente a la máquina de estado tratada referente a la transmisión serial

Desarrollo por Suma de Productos:

$$\begin{aligned}
 D_0 (Q_1, Q_0, Rx, Cuenta1, Cuenta2) = & (\overline{Q_1})(\overline{Q_0})(Rx = 0) + (\overline{Q_1})(Q_0)(\overline{Cuenta1} = 217) + \\
 & +(Q_1)(\overline{Q_0})(Cuenta2 = 9) + (Q_1)(\overline{Q_0})(Cuenta1 = 434)(\overline{Cuenta2} = 9)
 \end{aligned}$$

$$\begin{aligned}
 D_1 (Rx, Cuenta, Cuenta2) = & (\overline{Q_1})(Q_0)(Cuenta1 = 217) + (Q_1)(\overline{Q_0})(Cuenta1 \neq 434) + \\
 & +(Q_1)(\overline{Q_0})(Cuenta2 = 9)
 \end{aligned}$$

5. Primer, Segundo y Tercer Byte Recibido: Direccionamiento, Comando y Datos

El byte de dirección tiene la función de indicar a cuáles máquinas se envía la información de los otros dos bytes próximos, es decir, el byte comando y el byte de datos.

La dirección puede indicarse en forma individual, es decir, a una mezcladora en particular; a un grupo de mezcladoras previamente configurado, o por último, en modo *broadcast*, es decir, a todas las mezcladoras a la vez. Tanto la indicación individual como la de *broadcast* salen directamente del byte de datos, mientras que el direccionamiento por grupo es almacenado en la entidad comandar y entrada del bloque direccionar, dichos grupos son los configurados a los cuales la transmisión se envía.

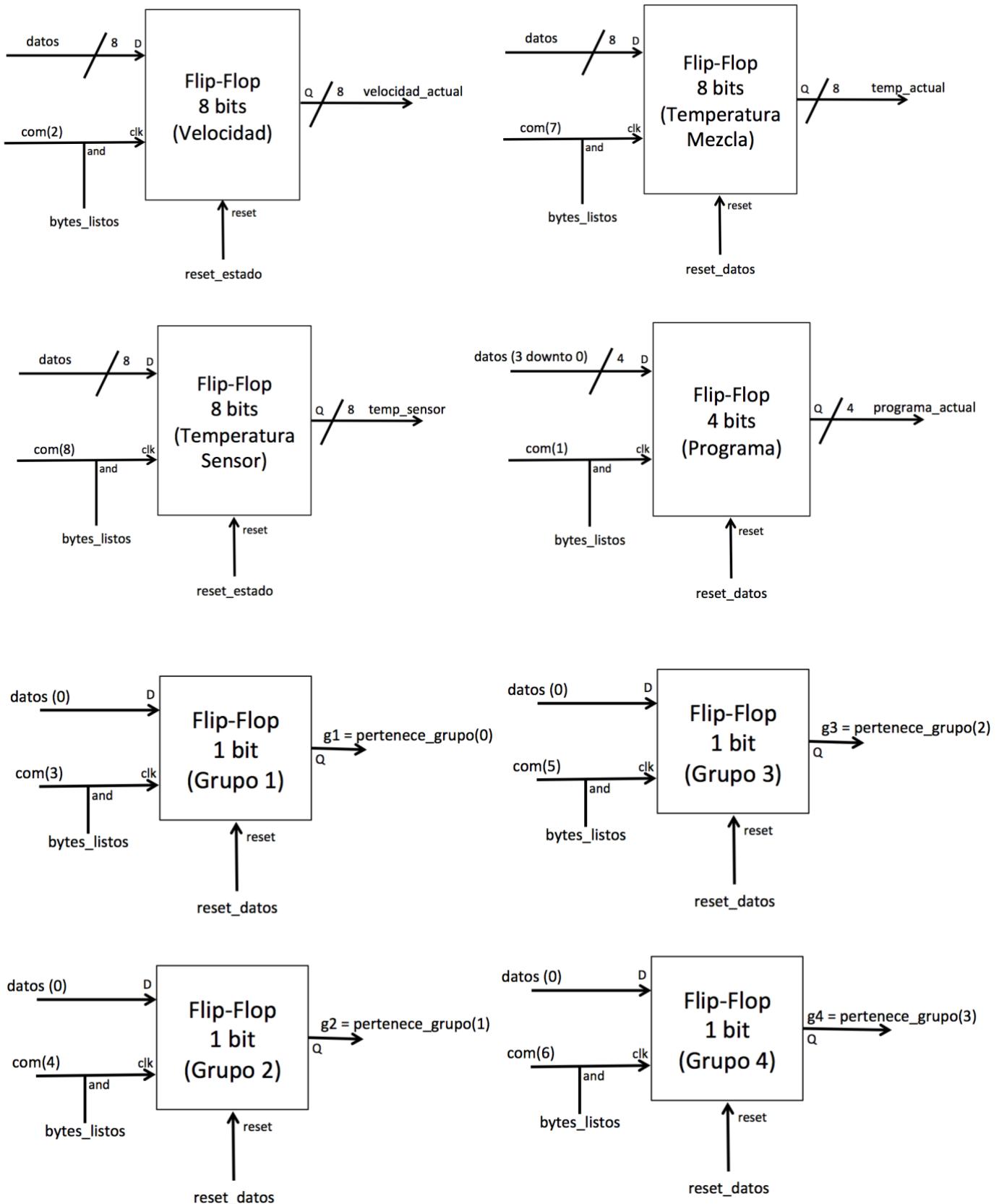
Según el formato del byte de dirección, se contempló que basta con algunas funciones lógicas básicas (compuertas *and* y *or*) para su decodificación.

Dado que el byte de datos está intrínsecamente asociado al byte de comando, se decidió por el procesamiento de ambos bytes en conjunto.

Los comandos indican qué función debe realizarse en la mezcladora previamente indicada (byte de dirección), ya sea para que configurar un programa, ejecutarlo, se configure la velocidad de mezclado, los grupos a los que pertenece, la temperatura de la mezcla, se envíe el valor de la temperatura obtenido de un sensor, o se le ordene una parada de emergencia.

Los datos indican la información requerida por el comando enviado. Ya sea un número de programa, un valor para la temperatura o velocidad angular o para indicar si la mezcladora debe o no pertenecer a cierto grupo.

Mediante un decodificador se obtienen las señales independientes de cada comando, luego con ellas se analizan los datos y se almacenan en flip flops, estos datos representan los datos actuales (configurados) de la mezcladora.



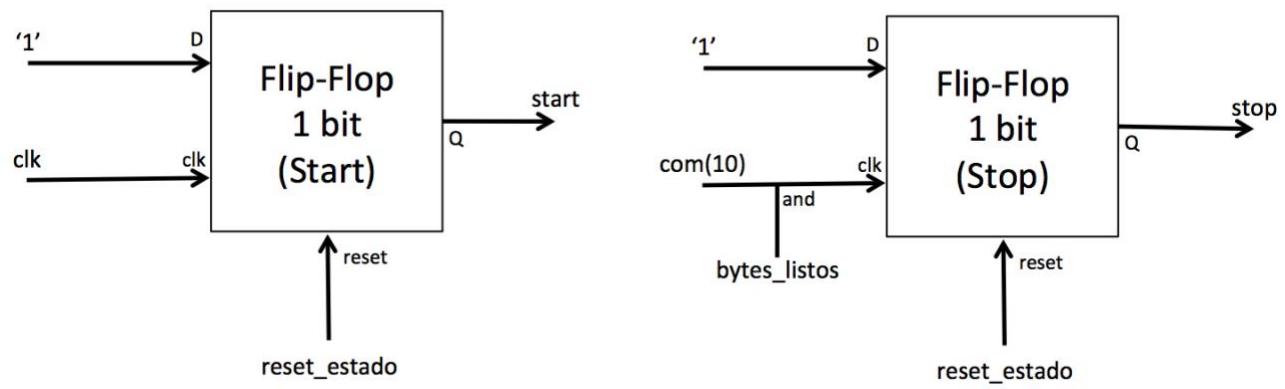


Figura 7. Diagrama de bloques de los Flip-Flops utilizados en el bloque “comandar” con sus respectivas entradas y salidas

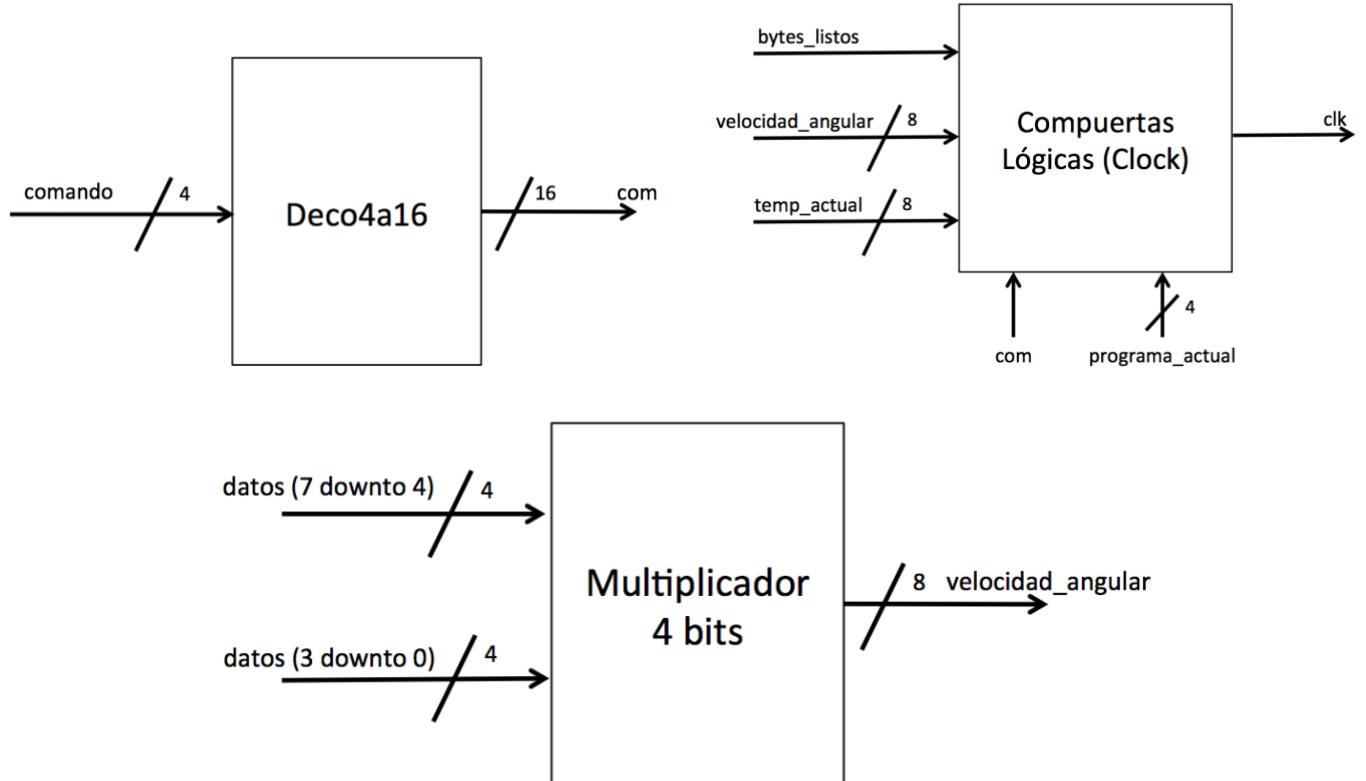


Figura 8. Diagrama de bloques de programas auxiliares utilizados en el bloque “comandar” con sus respectivas entradas y salidas

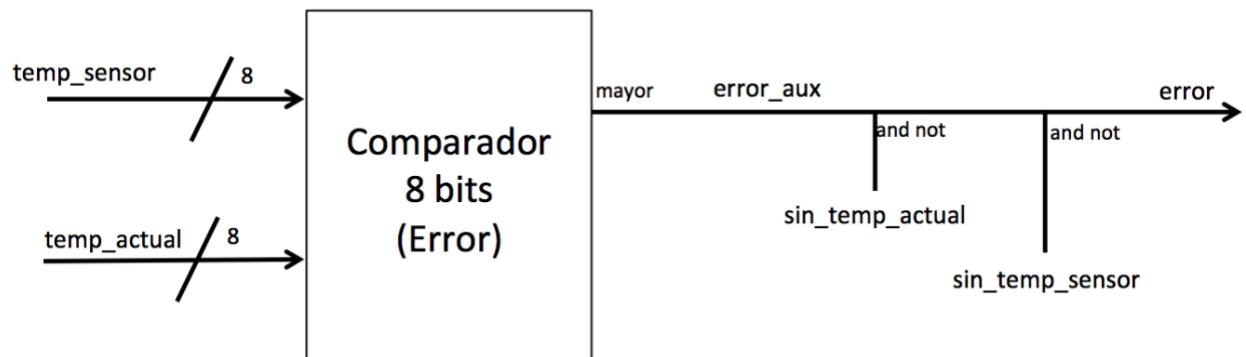
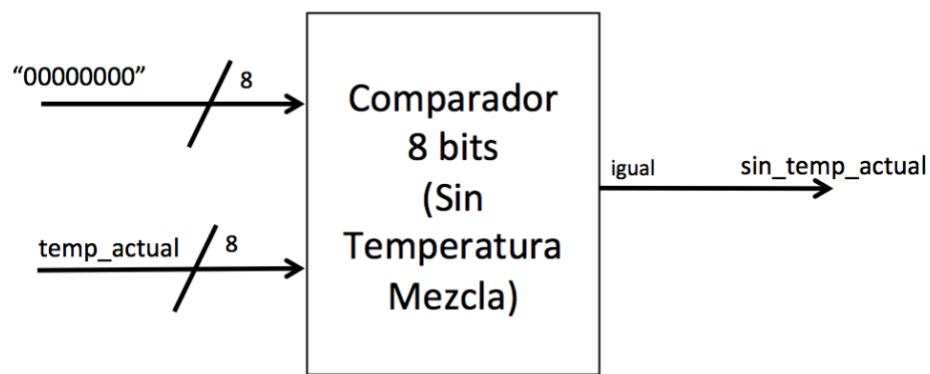
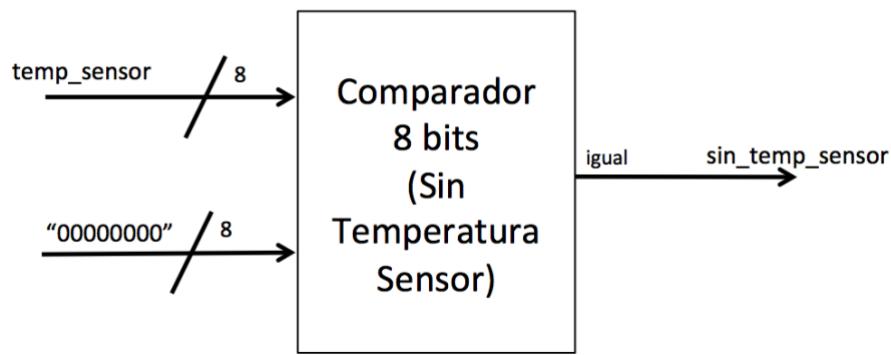


Figura 9. Diagrama de bloques de los comparadores utilizados en el bloque “comandar” con sus respectivas entradas y salidas

6. Codificación de los Programas Solicitados

6.1. Contadores Concatenados

Para realizar un contador se prosiguió a programar el circuito planteado por los docentes en el Laboratorio 2 de VHDL, consistente en dos multiplexores de 4 bits de 2 entradas y 1 salida en cascada junto a un flip-flop tipo D también de 4 bits. De esta forma, un ejemplo del diagrama de bloques de un contador simple y básico de 4 bits resultaría como lo esquematizado a continuación.

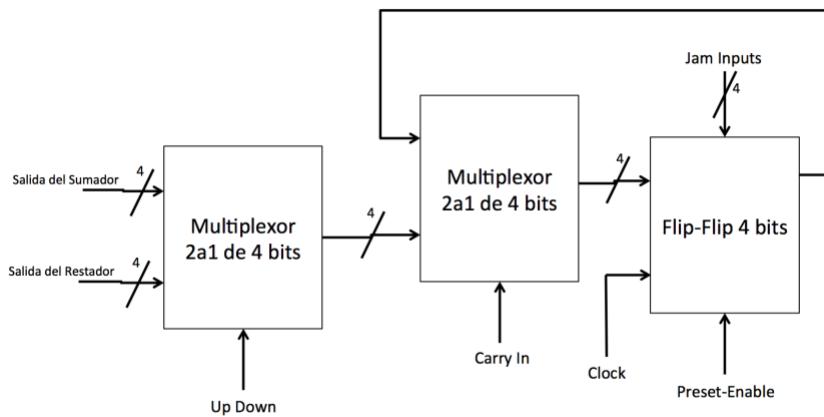


Figura 10. Diagrama de bloques de la estructura interna de un contador programado en VHDL

Como se puede observar, el primer multiplexor (ubicado a la izquierda de la figura) resta o suma a la cuenta recabada si la señal del bit de control es 0 o 1 respectivamente, cumpliendo por tanto la función del *Up/Down* de un contador como el trabajado en el Obligatorio 1 (CD4029). De la misma manera, el segundo multiplexor determina si el contador congela o prosigue la cuenta (al multiplexar a la salida del flip-flop o a la salida del multiplexor anterior para la continuación del proceso) determinado por el bit de control que de esta forma cumple el rol del *Carry In* de un contador básico, variando al tener señal lógica 0 o 1 respectivamente. Por último, el flip-flop posee como entradas la salida del segundo multiplexor, el cual no variará respecto a la señal actual en caso que el *Carry In* posea señal lógica 0 o variando en caso contrario, aprovechando de esta forma las propiedades características analizadas en el Obligatorio 1 de los flip-flop. Lógicamente, el reset del flip-flop mencionado funcionará como el *Preset-Enable* del contador, reiniciando la cuenta en caso de poseer este señal lógica 0 y funcionando normalmente en caso de poseer señal lógica 1. Sin embargo, este

reinicio de la cuenta solo se genera en caso de tener las *Jam Inputs* a Tierra (señal lógica 0) puesto que como ya fue visto el *Preset-Enable* se encarga de trasladar las entradas a las salidas del contador (es decir $(Q_3, Q_2, Q_1, Q_0) = (J_3, J_2, J_1, J_0)$). Estas *Jam Inputs* se encuentran por tanto conectadas al Set del flip-flop (teóricamente, ya que los flip-flops utilizados, programados, no poseen dichas entradas sino que poseen una condición interna que resetea el flip-flop), coherente nuevamente con la funcionalidad del flip-flop.

Como se muestra en la imagen siguiente, se concatenaron varios contadores de tal forma de generar el contador solicitado que cuente en décimas, unidades y decenas de segundos para su posterior muestra en la pantalla 1, se genera mediante la unión de varios contadores simples de los descritos anteriormente en cascada, unidos entre sí por el *Carry Out* del anterior al *Clock* del siguiente. Esta concatenación es lógica al considerar que el *Carry Out* posee valor lógico 1 al momento de resetearse la cuenta del contador para luego volver a poseer una señal lógica 0. Este cambio de señal es la que genera que el *clock* tenga flancos en el momento en que se activa el *Carry Out* y de esta manera, al fijar que la cuenta se resetee al llegar al número decimal 10 (en binario 1010) provoca que el reloj del segundo contador (aquel que cuenta las unidades de segundo transcurridas) sea de 1 Hz (1 segundo de período) y análogamente con el tercer contador (encargado de contar las decenas de segundo transcurridas), al tener de esta manera un reloj de 0.1 Hz (10 segundos de período). Por último, el contador de las decenas se resetea automáticamente al llegar a 6 ya que 60 segundos es la duración máxima de un programa.

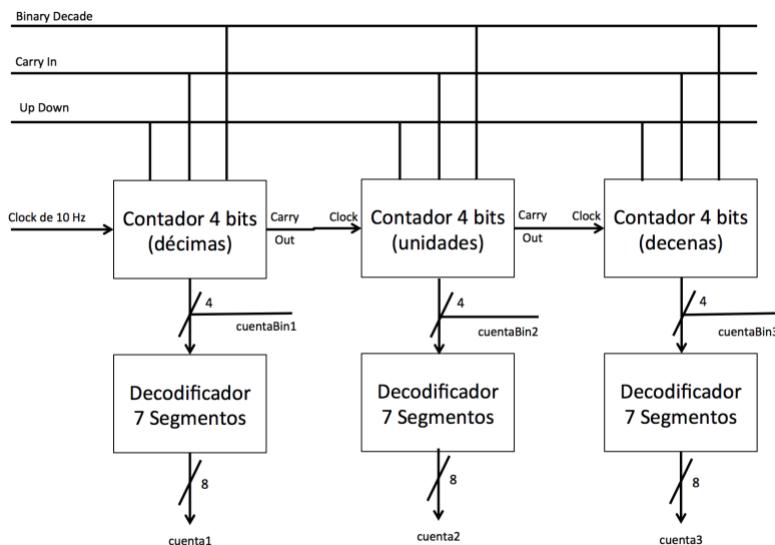


Figura 11. Diagrama de bloques del contador decimal necesitado creado en base a instancias de diferentes contadores

6.2. Programa 1

Según lo planteado en la letra del Obligatorio 2, la máquina de estados del Programa 1 consiste en 9 estados, correspondientes al uso de 4 flip-flops de 1 bit o 1 flip flop de 4 bits, representados en forma cíclica según el diagrama de la máquina de estados de la figura siguiente.

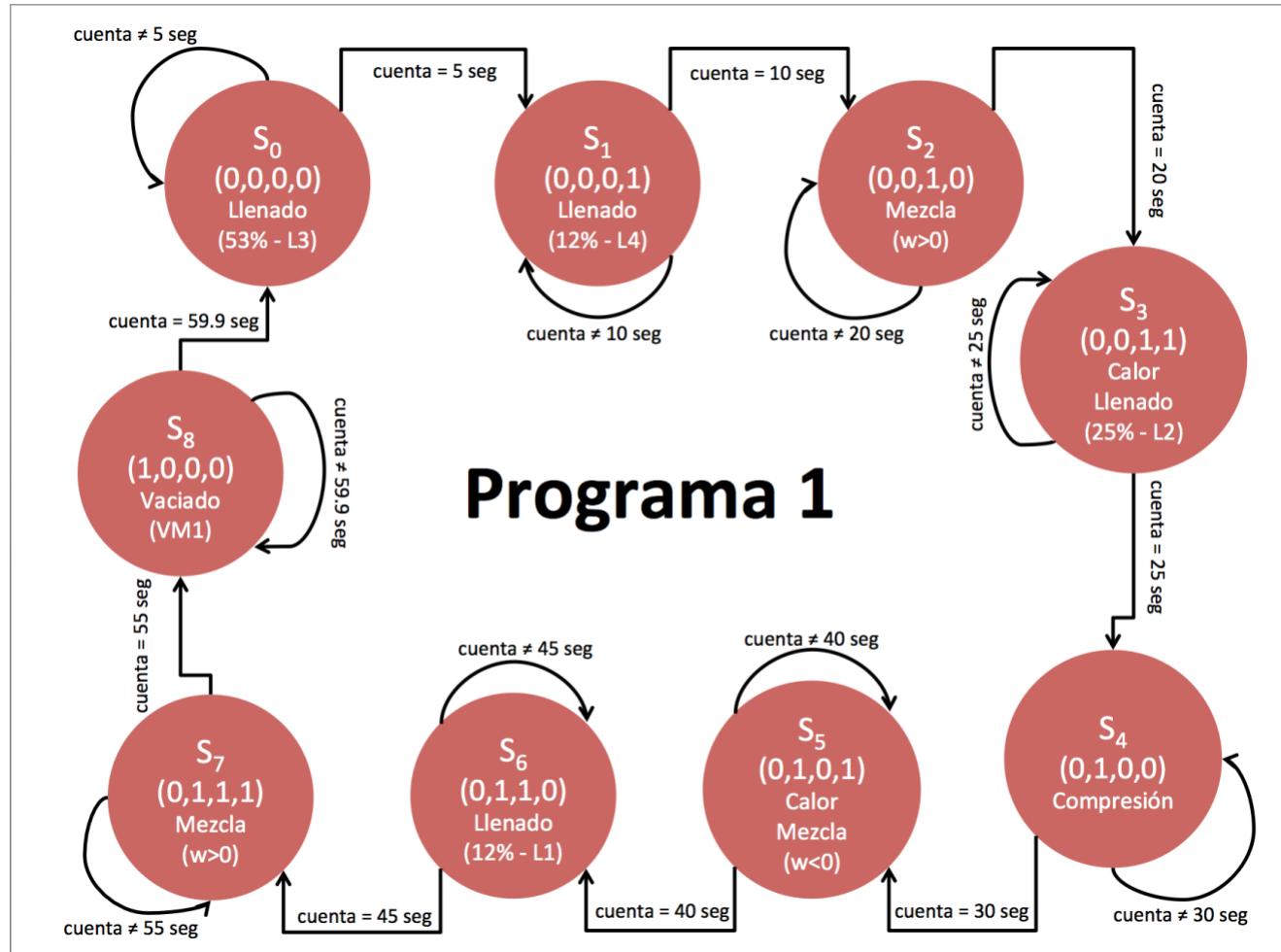


Figura 12: Máquina de Estados del Programa 1

La tabla correspondiente a la máquina de estados anterior se muestra en la tabla adjunta a continuación. Nótese que se usaron flip-flop tipo D ya que fueron los recomendados para su uso en este trabajo.

Estado Actual					Entradas del Flip-Flop tipo D				Condición	Estado Siguiente				
S_i	$Q_3(i)$	$Q_2(i)$	$Q_1(i)$	$Q_0(i)$	D_3	D_2	D_1	D_0		S_{i+1}	$Q_3(i+1)$	$Q_2(i+1)$	$Q_1(i+1)$	$Q_0(i+1)$
S_0	0	0	0	0	0	0	0	0	$cuenta \neq 5$	S_0	0	0	0	0
S_0	0	0	0	0	0	0	0	1	$cuenta = 5$	S_1	0	0	0	1
S_1	0	0	0	1	0	0	0	1	$cuenta \neq 10$	S_1	0	0	0	1
S_1	0	0	0	1	0	0	1	0	$cuenta = 10$	S_2	0	0	1	0
S_2	0	0	1	0	0	0	1	0	$cuenta \neq 20$	S_2	0	0	1	0
S_2	0	0	1	0	0	0	1	1	$cuenta = 20$	S_3	0	0	1	1
S_3	0	0	1	1	0	0	1	1	$cuenta \neq 25$	S_3	0	0	1	1
S_3	0	0	1	1	0	1	0	0	$cuenta = 25$	S_4	0	1	0	0
S_4	0	1	0	0	0	1	0	0	$cuenta \neq 30$	S_4	0	1	0	0
S_4	0	1	0	0	0	1	0	1	$cuenta = 30$	S_5	0	1	0	1
S_5	0	1	0	1	0	1	0	1	$cuenta \neq 40$	S_5	0	1	0	1
S_5	0	1	0	1	0	1	1	0	$cuenta = 40$	S_6	0	1	1	0
S_6	0	1	1	0	0	1	1	0	$cuenta \neq 45$	S_6	0	1	1	0
S_6	0	1	1	0	0	1	1	1	$cuenta = 45$	S_7	0	1	1	1
S_7	0	1	1	1	0	1	1	1	$cuenta \neq 55$	S_7	0	1	1	1
S_7	0	1	1	1	1	0	0	0	$cuenta = 55$	S_8	1	0	0	0
S_8	1	0	0	0	1	0	0	0	$cuenta \neq 59.9$	S_8	1	0	0	0
S_8	1	0	0	0	0	0	0	0	$cuenta = 59.9$	S_0	0	0	0	0

Estado Actual					Entradas del Flip-Flop tipo D				Condición	Estado Siguiente				
S_i	$Q_3(i)$	$Q_2(i)$	$Q_1(i)$	$Q_0(i)$	D_3	D_2	D_1	D_0		S_{i+1}	$Q_3(i+1)$	$Q_2(i+1)$	$Q_1(i+1)$	$Q_0(i+1)$
S_9	1	0	0	1	0	0	0	0	x	S_0	0	0	0	0
S_{10}	1	0	1	0	0	0	0	0	x	S_0	0	0	0	0
S_{11}	1	0	1	1	0	0	0	0	x	S_0	0	0	0	0
S_{12}	1	1	0	0	0	0	0	0	x	S_0	0	0	0	0
S_{13}	1	1	0	1	0	0	0	0	x	S_0	0	0	0	0
S_{14}	1	1	1	0	0	0	0	0	x	S_0	0	0	0	0
S_{15}	1	1	1	1	0	0	0	0	x	S_0	0	0	0	0

Tabla 3. Tabla de Verdad o Lógica referente a la máquina de estado tratada referente al Programa 1

Desarrollo por Suma de Productos:

$$\begin{aligned}
 D_0(Q_3, Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_3})(\overline{Q_2})(\overline{Q_1})(\overline{Q_0})(cuenta = 5) + (\overline{Q_3})(\overline{Q_2})(\overline{Q_1})(Q_0)(\overline{cuenta = 10}) + \\
 & + (\overline{Q_3})(\overline{Q_2})(Q_1)(\overline{Q_0})(cuenta = 20) + (\overline{Q_3})(\overline{Q_2})(Q_1)(Q_0)(\overline{cuenta = 25}) + \\
 & + (\overline{Q_3})(Q_2)(\overline{Q_1})(\overline{Q_0})(cuenta = 30) + (\overline{Q_3})(Q_2)(\overline{Q_1})(Q_0)(\overline{cuenta = 40}) + \\
 & + (\overline{Q_3})(Q_2)(Q_1)(\overline{Q_0})(cuenta = 45) + (\overline{Q_3})(Q_2)(Q_1)(Q_0)(\overline{cuenta = 55})
 \end{aligned}$$

$$\begin{aligned}
 D_1(Q_3, Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_3})(\overline{Q_2})(\overline{Q_1})(Q_0)(cuenta = 10) + (\overline{Q_3})(\overline{Q_2})(Q_1)(\overline{Q_0})(\overline{cuenta = 20}) + \\
 & + (\overline{Q_3})(\overline{Q_2})(Q_1)(\overline{Q_0})(cuenta = 20) + (\overline{Q_3})(\overline{Q_2})(Q_1)(Q_0)(\overline{cuenta = 25}) + \\
 & + (\overline{Q_3})(Q_2)(\overline{Q_1})(Q_0)(cuenta = 40) + (\overline{Q_3})(Q_2)(Q_1)(\overline{Q_0})(cuenta = 45) + \\
 & + (\overline{Q_3})(Q_2)(Q_1)(\overline{Q_0})(\overline{cuenta = 45}) + (\overline{Q_3})(Q_2)(Q_1)(Q_0)(\overline{cuenta = 55})
 \end{aligned}$$

$$\begin{aligned}
D_2(Q_3, Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_3})(\overline{Q_2})(Q_1)(Q_0)(cuenta = 25) + (\overline{Q_3})(Q_2)(\overline{Q_1})(\overline{Q_0})(\overline{cuenta = 30}) + \\
& + (\overline{Q_3})(Q_2)(\overline{Q_1})(\overline{Q_0})(cuenta = 30) + (\overline{Q_3})(Q_2)(\overline{Q_1})(Q_0)(\overline{cuenta = 40}) + \\
& + (\overline{Q_3})(Q_2)(\overline{Q_1})(Q_0)(cuenta = 40) + (\overline{Q_3})(Q_2)(Q_1)(\overline{Q_0})(\overline{cuenta = 45}) + \\
& + (\overline{Q_3})(Q_2)(Q_1)(\overline{Q_0})(cuenta = 45) + (\overline{Q_3})(Q_2)(Q_1)(Q_0)(\overline{cuenta = 55})
\end{aligned}$$

$$D_3(Q_3, Q_2, Q_1, Q_0, cuenta) = (\overline{Q_3})(Q_2)(Q_1)(Q_0)(cuenta = 55) + (Q_3)(\overline{Q_2})(\overline{Q_1})(\overline{Q_0})(\overline{cuenta = 59.9})$$

Reduciendo por Identidad de la Lógica de Boole referida a $[a + \bar{a} = 1]$:

$$\begin{aligned}
D_1(Q_3, Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_3})(\overline{Q_2})(\overline{Q_1})(Q_0)(cuenta = 10) + (\overline{Q_3})(\overline{Q_2})(Q_1)(\overline{Q_0}) + \\
& + (\overline{Q_3})(\overline{Q_2})(Q_1)(Q_0)(\overline{cuenta = 25}) + (\overline{Q_3})(Q_2)(\overline{Q_1})(Q_0)(cuenta = 40) + \\
& + (\overline{Q_3})(Q_2)(Q_1)(\overline{Q_0}) + (\overline{Q_3})(Q_2)(Q_1)(Q_0)(\overline{cuenta = 55})
\end{aligned}$$

$$\begin{aligned}
D_2(Q_3, Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_3})(\overline{Q_2})(Q_1)(Q_0)(cuenta = 25) + (\overline{Q_3})(Q_2)(\overline{Q_1})(\overline{Q_0}) + \\
& + (\overline{Q_3})(Q_2)(\overline{Q_1})(Q_0) + (\overline{Q_3})(Q_2)(Q_1)(\overline{Q_0}) + (\overline{Q_3})(Q_2)(Q_1)(Q_0)(\overline{cuenta = 55})
\end{aligned}$$

6.3. Programa 2

De la misma manera, la máquina de estados del Programa 2 consiste en 8 estados, correspondientes al uso de 3 flip-flops de 1 bit o 1 flip flop de 3, representados en forma cíclica según el diagrama de la máquina de estados de la figura siguiente.

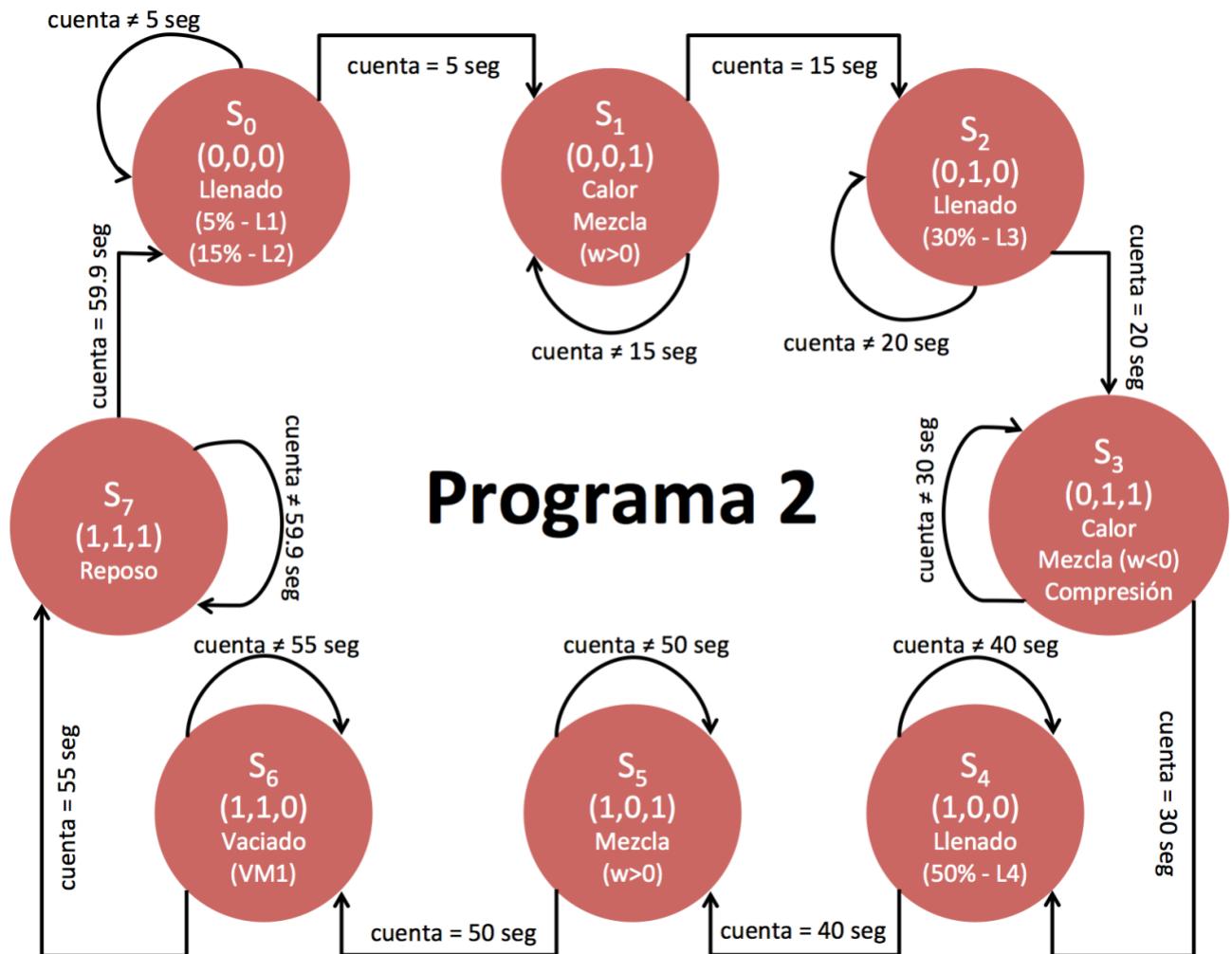


Figura 13: Máquina de Estados del Programa 2

La tabla correspondiente a la máquina de estados anterior se muestra en la tabla adjunta a continuación.

Estado Actual				Entradas de los Flip-Flop tipo D			Condición	Estado Siguiente			
S_i	$Q_2(i)$	$Q_1(i)$	$Q_0(i)$	D_2	D_1	D_0		S_{i+1}	$Q_2(i+1)$	$Q_1(i+1)$	$Q_0(i+1)$
S_0	0	0	0	0	0	0	$cuenta \neq 5$	S_0	0	0	0
S_0	0	0	0	0	0	1	$cuenta = 5$	S_1	0	0	1
S_1	0	0	1	0	0	1	$cuenta \neq 15$	S_1	0	0	1
S_1	0	0	1	0	1	0	$cuenta = 15$	S_2	0	1	0
S_2	0	1	0	0	1	0	$cuenta \neq 20$	S_2	0	1	0
S_2	0	1	0	0	1	1	$cuenta = 20$	S_3	0	1	1
S_3	0	1	1	0	1	1	$cuenta \neq 30$	S_3	0	1	1
S_3	0	1	1	1	0	0	$cuenta = 30$	S_4	1	0	0
S_4	1	0	0	1	0	0	$cuenta \neq 40$	S_4	1	0	0
S_4	1	0	0	1	0	1	$cuenta = 40$	S_5	1	0	1
S_5	1	0	1	1	0	1	$cuenta \neq 50$	S_5	1	0	1
S_5	1	0	1	1	1	0	$cuenta = 50$	S_6	1	1	0
S_6	1	1	0	1	1	0	$cuenta \neq 55$	S_6	1	1	0
S_6	1	1	0	1	1	1	$cuenta = 55$	S_7	1	1	1
S_7	1	1	1	1	1	1	$cuenta \neq 59.9$	S_7	1	1	1
S_7	1	1	1	0	0	0	$cuenta = 59.9$	S_0	0	0	0

Tabla 4. Tabla de Verdad o Lógica referente a la máquina de estado tratada referente al Programa 2

Desarrollo por Suma de Productos:

$$\begin{aligned}
 D_0(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(\overline{Q_1})(\overline{Q_0})(cuenta = 5) + (\overline{Q_2})(\overline{Q_1})(Q_0)(\overline{cuenta = 15}) + \\
 & + (\overline{Q_2})(Q_1)(\overline{Q_0})(cuenta = 20) + (\overline{Q_2})(Q_1)(Q_0)(\overline{cuenta = 30}) + \\
 & + (Q_2)(\overline{Q_1})(\overline{Q_0})(cuenta = 40) + (Q_2)(\overline{Q_1})(Q_0)(\overline{cuenta = 50}) + \\
 & + (Q_2)(Q_1)(\overline{Q_0})(cuenta = 55) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

$$\begin{aligned}
 D_1(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(\overline{Q_1})(Q_0)(cuenta = 15) + (\overline{Q_2})(Q_1)(\overline{Q_0})(\overline{cuenta = 20}) + \\
 & + (\overline{Q_2})(Q_1)(\overline{Q_0})(cuenta = 20) + (\overline{Q_2})(Q_1)(Q_0)(\overline{cuenta = 30}) + \\
 & + (Q_2)(\overline{Q_1})(Q_0)(cuenta = 50) + (Q_2)(Q_1)(\overline{Q_0})(cuenta = 55) + \\
 & + (Q_2)(Q_1)(\overline{Q_0})(\overline{cuenta = 55}) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

$$\begin{aligned}
 D_2(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(Q_1)(Q_0)(cuenta = 30) + (Q_2)(\overline{Q_1})(\overline{Q_0})(\overline{cuenta = 40}) + \\
 & + (Q_2)(\overline{Q_1})(\overline{Q_0})(cuenta = 40) + (Q_2)(\overline{Q_1})(Q_0)(\overline{cuenta = 50}) + \\
 & + (Q_2)(\overline{Q_1})(Q_0)(cuenta = 50) + (Q_2)(Q_1)(\overline{Q_0})(\overline{cuenta = 55}) + \\
 & + (Q_2)(Q_1)(\overline{Q_0})(cuenta = 55) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

Reduciendo por Identidad de la Lógica de Boole referida a $[a + \bar{a} = 1]$:

$$\begin{aligned}
 D_1(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(\overline{Q_1})(Q_0)(cuenta = 15) + (\overline{Q_2})(Q_1)(\overline{Q_0}) + \\
 & + (\overline{Q_2})(Q_1)(Q_0)(\overline{cuenta = 30}) + (Q_2)(\overline{Q_1})(Q_0)(cuenta = 50) + \\
 & + (Q_2)(Q_1)(\overline{Q_0}) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

$$\begin{aligned}
 D_2(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(Q_1)(Q_0)(cuenta = 30) + (Q_2)(\overline{Q_1})(\overline{Q_0}) + \\
 & + (Q_2)(\overline{Q_1})(Q_0) + (Q_2)(Q_1)(\overline{Q_0}) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

6.4. Programa 3

Asimismo, la máquina de estados del Programa 3 consiste en 8 estados, correspondientes al uso de 3 flip-flops de 1 bit o 1 flip flop de 3, representados en forma cíclica según el diagrama de la máquina de estados de la figura siguiente.

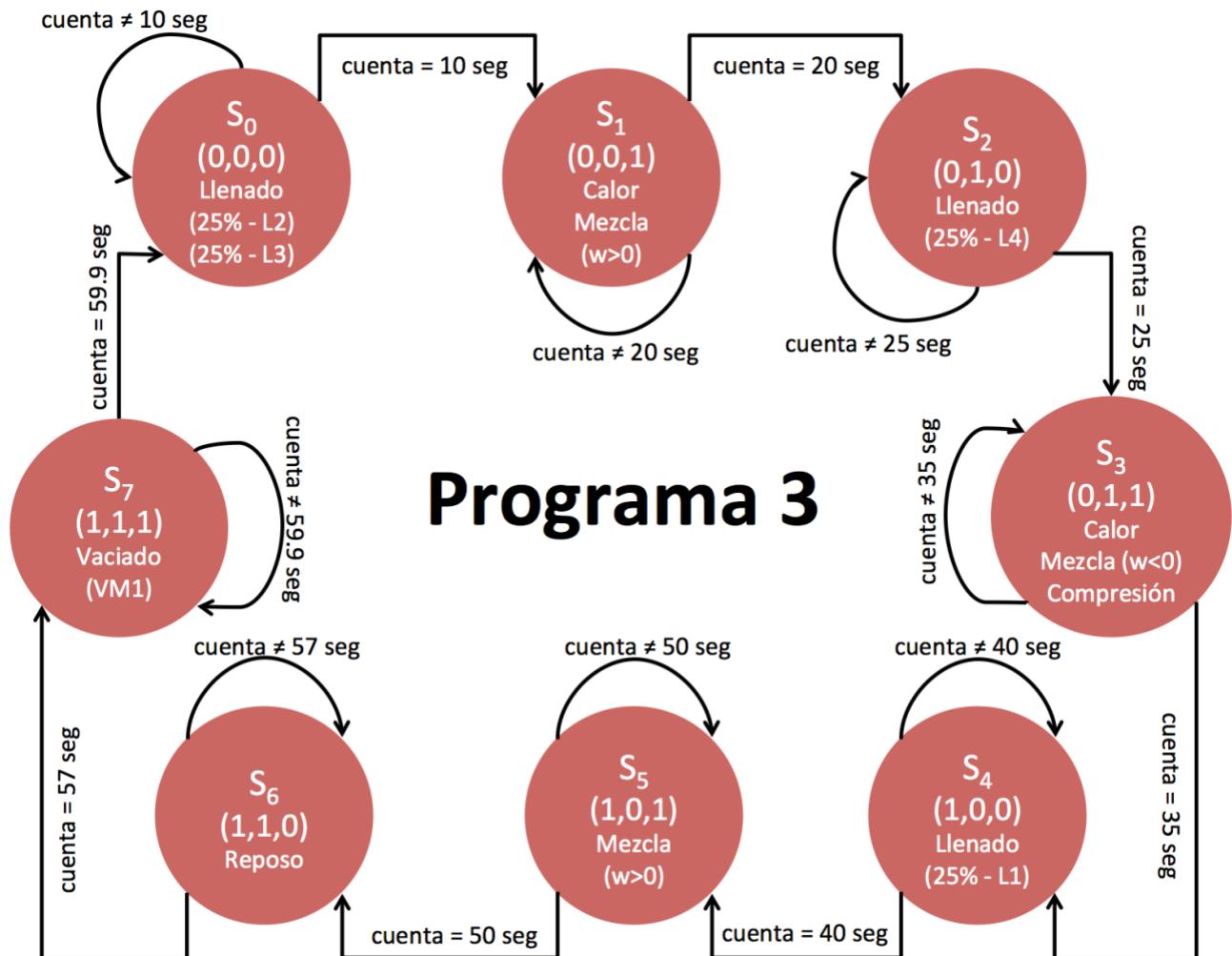


Figura 14: Máquina de Estados del Programa 3

La tabla correspondiente a la máquina de estados anterior se muestra en la tabla adjunta a continuación.

Estado Actual				Entradas de los Flip-Flop tipo D			Condición	Estado Siguiente			
S_i	$Q_2(i)$	$Q_1(i)$	$Q_0(i)$	D_2	D_1	D_0		S_{i+1}	$Q_2(i+1)$	$Q_1(i+1)$	$Q_0(i+1)$
S_0	0	0	0	0	0	0	$cuenta \neq 10$	S_0	0	0	0
S_0	0	0	0	0	0	1	$cuenta = 10$	S_1	0	0	1
S_1	0	0	1	0	0	1	$cuenta \neq 20$	S_1	0	0	1
S_1	0	0	1	0	1	0	$cuenta = 20$	S_2	0	1	0
S_2	0	1	0	0	1	0	$cuenta \neq 25$	S_2	0	1	0
S_2	0	1	0	0	1	1	$cuenta = 25$	S_3	0	1	1
S_3	0	1	1	0	1	1	$cuenta \neq 35$	S_3	0	1	1
S_3	0	1	1	1	0	0	$cuenta = 35$	S_4	1	0	0
S_4	1	0	0	1	0	0	$cuenta \neq 40$	S_4	1	0	0
S_4	1	0	0	1	0	1	$cuenta = 40$	S_5	1	0	1
S_5	1	0	1	1	0	1	$cuenta \neq 50$	S_5	1	0	1
S_5	1	0	1	1	1	0	$cuenta = 50$	S_6	1	1	0
S_6	1	1	0	1	1	0	$cuenta \neq 57$	S_6	1	1	0
S_6	1	1	0	1	1	1	$cuenta = 57$	S_7	1	1	1
S_7	1	1	1	1	1	1	$cuenta \neq 59.9$	S_7	1	1	1
S_7	1	1	1	0	0	0	$cuenta = 59.9$	S_0	0	0	0

Tabla 5. Tabla de Verdad o Lógica referente a la máquina de estado tratada referente al Programa 1

Desarrollo por Suma de Productos:

$$\begin{aligned}
 D_0(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(\overline{Q_1})(\overline{Q_0})(cuenta = 10) + (\overline{Q_2})(\overline{Q_1})(Q_0)(\overline{cuenta = 20}) + \\
 & + (\overline{Q_2})(Q_1)(\overline{Q_0})(cuenta = 25) + (\overline{Q_2})(Q_1)(Q_0)(\overline{cuenta = 35}) + \\
 & + (Q_2)(\overline{Q_1})(\overline{Q_0})(cuenta = 40) + (Q_2)(\overline{Q_1})(Q_0)(\overline{cuenta = 50}) + \\
 & + (Q_2)(Q_1)(\overline{Q_0})(cuenta = 57) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

$$\begin{aligned}
 D_1(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(\overline{Q_1})(Q_0)(cuenta = 20) + (\overline{Q_2})(Q_1)(\overline{Q_0})(\overline{cuenta = 25}) + \\
 & + (\overline{Q_2})(Q_1)(\overline{Q_0})(cuenta = 25) + (\overline{Q_2})(Q_1)(Q_0)(\overline{cuenta = 35}) + \\
 & + (Q_2)(\overline{Q_1})(Q_0)(cuenta = 50) + (Q_2)(Q_1)(\overline{Q_0})(cuenta = 57) + \\
 & + (Q_2)(Q_1)(\overline{Q_0})(\overline{cuenta = 57}) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

$$\begin{aligned}
 D_2(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(Q_1)(Q_0)(cuenta = 35) + (Q_2)(\overline{Q_1})(\overline{Q_0})(\overline{cuenta = 40}) + \\
 & + (Q_2)(\overline{Q_1})(\overline{Q_0})(cuenta = 40) + (Q_2)(\overline{Q_1})(Q_0)(\overline{cuenta = 50}) + \\
 & + (Q_2)(\overline{Q_1})(Q_0)(cuenta = 50) + (Q_2)(Q_1)(\overline{Q_0})(\overline{cuenta = 57}) + \\
 & + (Q_2)(Q_1)(\overline{Q_0})(cuenta = 57) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

Reduciendo por Identidad de la Lógica de Boole referida a $[a + \bar{a} = 1]$:

$$\begin{aligned}
 D_1(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(\overline{Q_1})(Q_0)(cuenta = 20) + (\overline{Q_2})(Q_1)(\overline{Q_0}) + \\
 & + (\overline{Q_2})(Q_1)(Q_0)(\overline{cuenta = 30}) + (Q_2)(\overline{Q_1})(Q_0)(cuenta = 50) + \\
 & + (Q_2)(Q_1)(\overline{Q_0}) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

$$\begin{aligned}
 D_2(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(Q_1)(Q_0)(cuenta = 35) + (Q_2)(\overline{Q_1})(\overline{Q_0}) + \\
 & + (Q_2)(\overline{Q_1})(Q_0) + (Q_2)(Q_1)(\overline{Q_0}) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

6.5. Programa 4

De esta forma, la máquina de estados del Programa 4 consiste en 8 estados, correspondientes al uso de 3 flip-flops de 1 bit o 1 flip flop de 3, representados en forma cíclica según el diagrama de la máquina de estados de la figura siguiente.

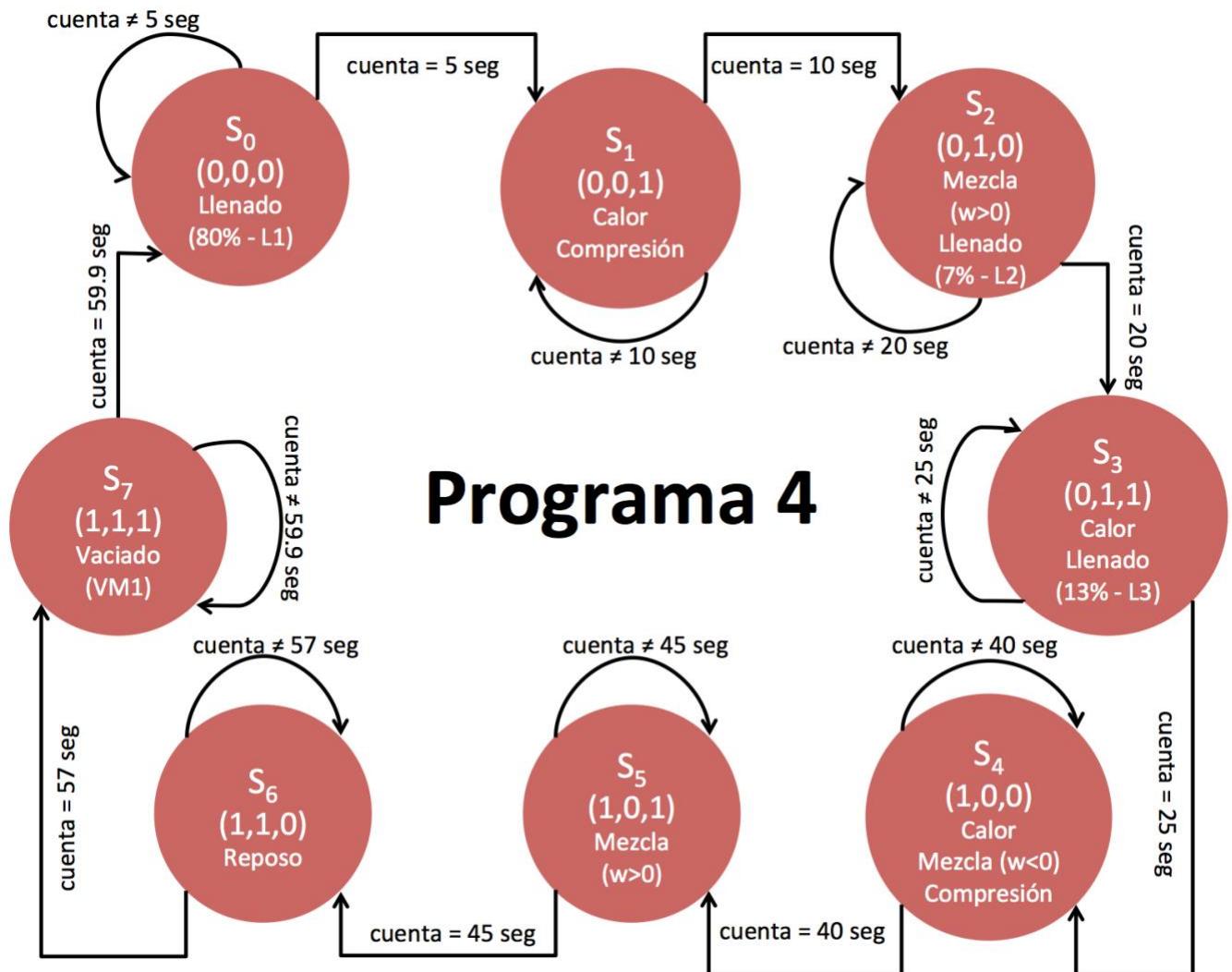


Figura 15: Máquina de Estados del Programa 4

La tabla correspondiente a la máquina de estados anterior se muestra en la tabla adjunta a continuación.

Estado Actual				Entradas de los Flip-Flop tipo D			Condición	Estado Siguiente			
S_i	$Q_2(i)$	$Q_1(i)$	$Q_0(i)$	D_2	D_1	D_0		S_{i+1}	$Q_2(i+1)$	$Q_1(i+1)$	$Q_0(i+1)$
S_0	0	0	0	0	0	0	$cuenta \neq 5$	S_0	0	0	0
S_0	0	0	0	0	0	1	$cuenta = 5$	S_1	0	0	1
S_1	0	0	1	0	0	1	$cuenta \neq 10$	S_1	0	0	1
S_1	0	0	1	0	1	0	$cuenta = 10$	S_2	0	1	0
S_2	0	1	0	0	1	0	$cuenta \neq 20$	S_2	0	1	0
S_2	0	1	0	0	1	1	$cuenta = 20$	S_3	0	1	1
S_3	0	1	1	0	1	1	$cuenta \neq 25$	S_3	0	1	1
S_3	0	1	1	1	0	0	$cuenta = 25$	S_4	1	0	0
S_4	1	0	0	1	0	0	$cuenta \neq 40$	S_4	1	0	0
S_4	1	0	0	1	0	1	$cuenta = 40$	S_5	1	0	1
S_5	1	0	1	1	0	1	$cuenta \neq 45$	S_5	1	0	1
S_5	1	0	1	1	1	0	$cuenta = 45$	S_6	1	1	0
S_6	1	1	0	1	1	0	$cuenta \neq 57$	S_6	1	1	0
S_6	1	1	0	1	1	1	$cuenta = 57$	S_7	1	1	1
S_7	1	1	1	1	1	1	$cuenta \neq 59.9$	S_7	1	1	1
S_7	1	1	1	0	0	0	$cuenta = 59.9$	S_0	0	0	0

Tabla 6. Tabla de Verdad o Lógica referente a la máquina de estado tratada referente al Programa 4

Desarrollo por Suma de Productos:

$$D_0(Q_2, Q_1, Q_0, cuenta) = (\overline{Q_2})(\overline{Q_1})(\overline{Q_0})(cuenta = 5) + (\overline{Q_2})(\overline{Q_1})(Q_0)(\overline{cuenta = 10}) + \\ + (\overline{Q_2})(Q_1)(\overline{Q_0})(cuenta = 20) + (\overline{Q_2})(Q_1)(Q_0)(\overline{cuenta = 25}) + \\ + (Q_2)(\overline{Q_1})(\overline{Q_0})(cuenta = 40) + (Q_2)(\overline{Q_1})(Q_0)(\overline{cuenta = 45}) + \\ + (Q_2)(Q_1)(\overline{Q_0})(cuenta = 57) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})$$

$$D_1(Q_2, Q_1, Q_0, cuenta) = (\overline{Q_2})(\overline{Q_1})(Q_0)(cuenta = 10) + (\overline{Q_2})(Q_1)(\overline{Q_0})(cuenta = 20) + \\ + (\overline{Q_2})(Q_1)(\overline{Q_0})(\overline{cuenta = 20}) + (\overline{Q_2})(Q_1)(Q_0)(\overline{cuenta = 25}) + \\ + (Q_2)(\overline{Q_1})(Q_0)(cuenta = 45) + (Q_2)(Q_1)(\overline{Q_0})(cuenta = 57) + \\ + (Q_2)(Q_1)(\overline{Q_0})(\overline{cuenta = 57}) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})$$

$$D_2(Q_2, Q_1, Q_0, cuenta) = (\overline{Q_2})(Q_1)(Q_0)(cuenta = 25) + (Q_2)(\overline{Q_1})(\overline{Q_0})(\overline{cuenta = 40}) + \\ + (Q_2)(\overline{Q_1})(\overline{Q_0})(cuenta = 40) + (Q_2)(\overline{Q_1})(Q_0)(\overline{cuenta = 45}) + \\ + (Q_2)(\overline{Q_1})(Q_0)(cuenta = 45) + (Q_2)(Q_1)(\overline{Q_0})(\overline{cuenta = 57}) + \\ + (Q_2)(Q_1)(\overline{Q_0})(cuenta = 57) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})$$

Reduciendo por Identidad de la Lógica de Boole referida a $[a + \bar{a} = 1]$:

$$D_1(Q_2, Q_1, Q_0, cuenta) = (\overline{Q_2})(\overline{Q_1})(Q_0)(cuenta = 10) + (\overline{Q_2})(Q_1)(\overline{Q_0}) + \\ + (\overline{Q_2})(Q_1)(Q_0)(\overline{cuenta = 25}) + (Q_2)(\overline{Q_1})(Q_0)(cuenta = 45) + \\ + (Q_2)(Q_1)(\overline{Q_0}) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})$$

$$D_2(Q_2, Q_1, Q_0, cuenta) = (\overline{Q_2})(Q_1)(Q_0)(cuenta = 25) + (Q_2)(\overline{Q_1})(\overline{Q_0}) + \\ + (Q_2)(\overline{Q_1})(Q_0) + (Q_2)(Q_1)(\overline{Q_0}) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})$$

6.6. Programa 5

De igual manera, la máquina de estados del Programa 5 consiste en 8 estados, correspondientes al uso de 3 flip-flops de 1 bit o 1 flip flop de 3, representados en forma cíclica según el diagrama de la máquina de estados de la figura siguiente.

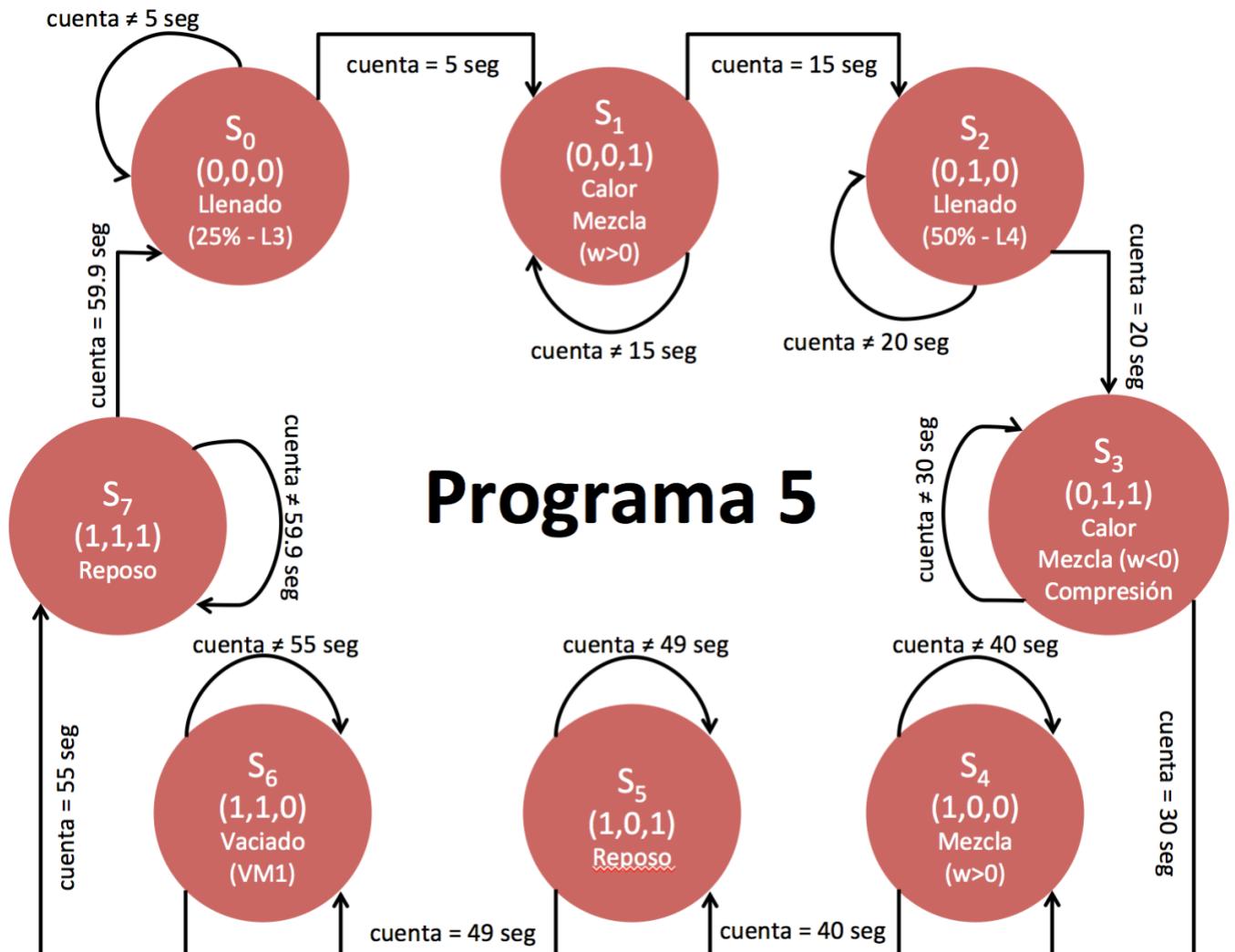


Figura 16: Máquina de Estados del Programa 5

La tabla correspondiente a la máquina de estados anterior se muestra en la tabla adjunta a continuación.

Estado Actual				Entradas de los Flip-Flop tipo D			Condición	Estado Siguiente			
S_i	$Q_2(i)$	$Q_1(i)$	$Q_0(i)$	D_2	D_1	D_0		S_{i+1}	$Q_2(i+1)$	$Q_1(i+1)$	$Q_0(i+1)$
S_0	0	0	0	0	0	0	$cuenta \neq 5$	S_0	0	0	0
S_0	0	0	0	0	0	1	$cuenta = 5$	S_1	0	0	1
S_1	0	0	1	0	0	1	$cuenta \neq 15$	S_1	0	0	1
S_1	0	0	1	0	1	0	$cuenta = 15$	S_2	0	1	0
S_2	0	1	0	0	1	0	$cuenta \neq 20$	S_2	0	1	0
S_2	0	1	0	0	1	1	$cuenta = 20$	S_3	0	1	1
S_3	0	1	1	0	1	1	$cuenta \neq 30$	S_3	0	1	1
S_3	0	1	1	1	0	0	$cuenta = 30$	S_4	1	0	0
S_4	1	0	0	1	0	0	$cuenta \neq 40$	S_4	1	0	0
S_4	1	0	0	1	0	1	$cuenta = 40$	S_5	1	0	1
S_5	1	0	1	1	0	1	$cuenta \neq 49$	S_5	1	0	1
S_5	1	0	1	1	1	0	$cuenta = 49$	S_6	1	1	0
S_6	1	1	0	1	1	0	$cuenta \neq 55$	S_6	1	1	0
S_6	1	1	0	1	1	1	$cuenta = 55$	S_7	1	1	1
S_7	1	1	1	1	1	1	$cuenta \neq 59.9$	S_7	1	1	1
S_7	1	1	1	0	0	0	$cuenta = 59.9$	S_0	0	0	0

Tabla 7. Tabla de Verdad o Lógica referente a la máquina de estado tratada referente al Programa 5

Desarrollo por Suma de Productos:

$$\begin{aligned}
 D_0(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(\overline{Q_1})(\overline{Q_0})(cuenta = 5) + (\overline{Q_2})(\overline{Q_1})(Q_0)(\overline{cuenta = 15}) + \\
 & + (\overline{Q_2})(Q_1)(\overline{Q_0})(cuenta = 20) + (\overline{Q_2})(Q_1)(Q_0)(\overline{cuenta = 30}) + \\
 & + (Q_2)(\overline{Q_1})(\overline{Q_0})(cuenta = 40) + (Q_2)(\overline{Q_1})(Q_0)(\overline{cuenta = 49}) + \\
 & + (Q_2)(Q_1)(\overline{Q_0})(cuenta = 55) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

$$\begin{aligned}
 D_1(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(\overline{Q_1})(Q_0)(cuenta = 15) + (\overline{Q_2})(Q_1)(\overline{Q_0})(cuenta = 20) + \\
 & + (\overline{Q_2})(Q_1)(\overline{Q_0})(\overline{cuenta = 20}) + (\overline{Q_2})(Q_1)(Q_0)(\overline{cuenta = 30}) + \\
 & + (Q_2)(\overline{Q_1})(Q_0)(cuenta = 49) + (Q_2)(Q_1)(\overline{Q_0})(cuenta = 55) + \\
 & + (Q_2)(Q_1)(\overline{Q_0})(\overline{cuenta = 55}) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

$$\begin{aligned}
 D_2(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(Q_1)(Q_0)(cuenta = 30) + (Q_2)(\overline{Q_1})(\overline{Q_0})(\overline{cuenta = 40}) + \\
 & + (Q_2)(\overline{Q_1})(\overline{Q_0})(cuenta = 40) + (Q_2)(\overline{Q_1})(Q_0)(\overline{cuenta = 49}) + \\
 & + (Q_2)(\overline{Q_1})(Q_0)(cuenta = 49) + (Q_2)(Q_1)(\overline{Q_0})(\overline{cuenta = 55}) + \\
 & + (Q_2)(Q_1)(\overline{Q_0})(cuenta = 55) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

Reduciendo por Identidad de la Lógica de Boole referida a $[a + \bar{a} = 1]$:

$$\begin{aligned}
 D_1(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(\overline{Q_1})(Q_0)(cuenta = 15) + (\overline{Q_2})(Q_1)(\overline{Q_0}) + \\
 & + (\overline{Q_2})(Q_1)(Q_0)(\overline{cuenta = 30}) + (Q_2)(\overline{Q_1})(Q_0)(cuenta = 49) + \\
 & + (Q_2)(Q_1)(\overline{Q_0}) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

$$\begin{aligned}
 D_2(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(Q_1)(Q_0)(cuenta = 30) + (Q_2)(\overline{Q_1})(\overline{Q_0}) + \\
 & + (Q_2)(\overline{Q_1})(Q_0) + (Q_2)(Q_1)(\overline{Q_0}) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

6.7. Programa 6

De la misma forma, la máquina de estados del Programa 6 consiste en 8 estados, correspondientes al uso de 3 flip-flops de 1 bit o 1 flip flop de 3, representados en forma cíclica según el diagrama de la máquina de estados de la figura siguiente.

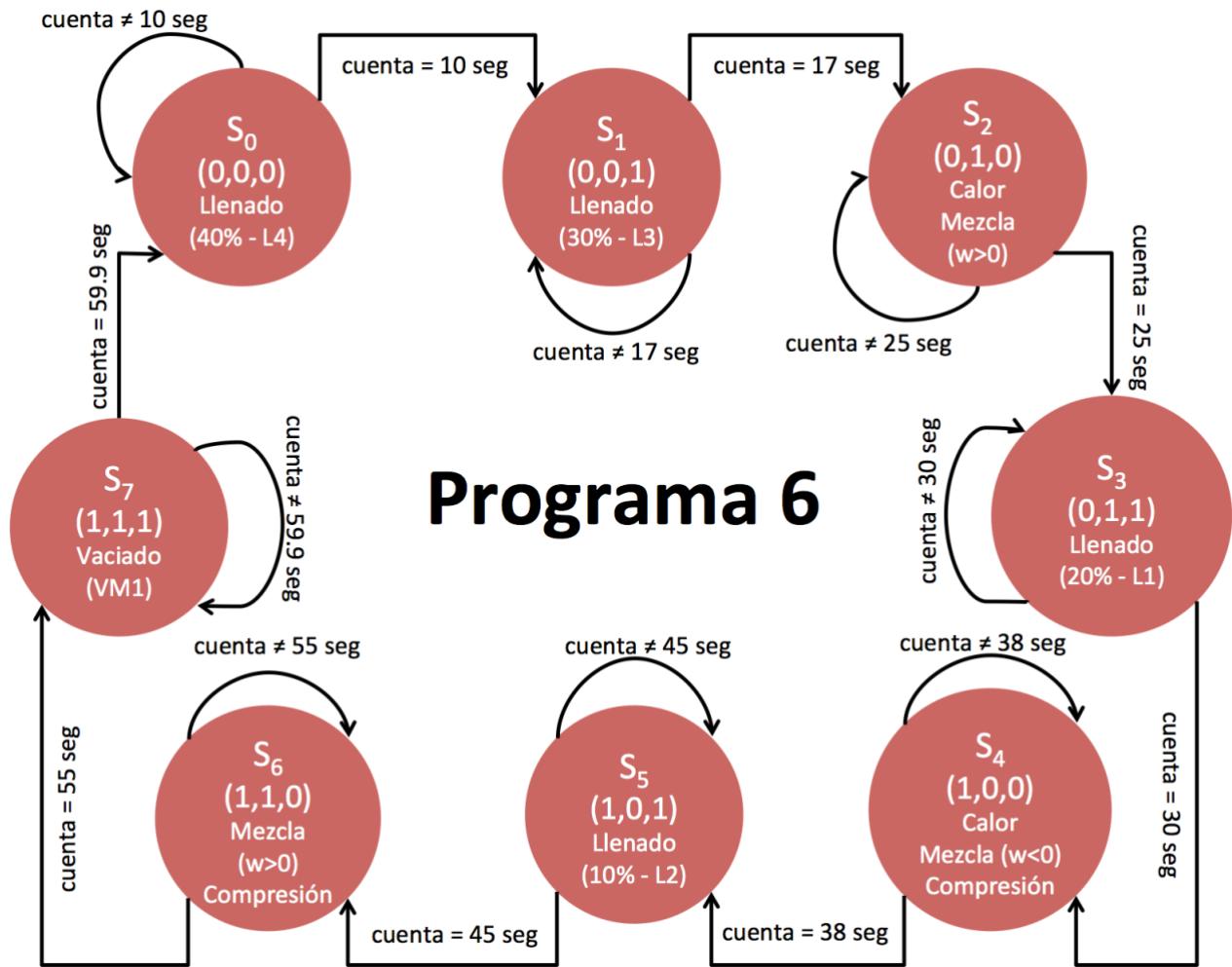


Figura 17: Máquina de Estados del Programa 6

La tabla correspondiente a la máquina de estados anterior se muestra en la tabla adjunta a continuación.

Estado Actual				Entradas de los Flip-Flop tipo D			Condición	Estado Siguiente			
S_i	$Q_2(i)$	$Q_1(i)$	$Q_0(i)$	D_2	D_1	D_0		S_{i+1}	$Q_2(i+1)$	$Q_1(i+1)$	$Q_0(i+1)$
S_0	0	0	0	0	0	0	$cuenta \neq 10$	S_0	0	0	0
S_0	0	0	0	0	0	1	$cuenta = 10$	S_1	0	0	1
S_1	0	0	1	0	0	1	$cuenta \neq 17$	S_1	0	0	1
S_1	0	0	1	0	1	0	$cuenta = 17$	S_2	0	1	0
S_2	0	1	0	0	1	0	$cuenta \neq 25$	S_2	0	1	0
S_2	0	1	0	0	1	1	$cuenta = 25$	S_3	0	1	1
S_3	0	1	1	0	1	1	$cuenta \neq 30$	S_3	0	1	1
S_3	0	1	1	1	0	0	$cuenta = 30$	S_4	1	0	0
S_4	1	0	0	1	0	0	$cuenta \neq 38$	S_4	1	0	0
S_4	1	0	0	1	0	1	$cuenta = 38$	S_5	1	0	1
S_5	1	0	1	1	0	1	$cuenta \neq 45$	S_5	1	0	1
S_5	1	0	1	1	1	0	$cuenta = 45$	S_6	1	1	0
S_6	1	1	0	1	1	0	$cuenta \neq 55$	S_6	1	1	0
S_6	1	1	0	1	1	1	$cuenta = 55$	S_7	1	1	1
S_7	1	1	1	1	1	1	$cuenta \neq 59.9$	S_7	1	1	1
S_7	1	1	1	0	0	0	$cuenta = 59.9$	S_0	0	0	0

Tabla 8. Tabla de Verdad o Lógica referente a la máquina de estado tratada referente al Programa 6

Desarrollo por Suma de Productos:

$$\begin{aligned}
 D_0(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(\overline{Q_1})(\overline{Q_0})(cuenta = 10) + (\overline{Q_2})(\overline{Q_1})(Q_0)(\overline{cuenta = 17}) + \\
 & +(\overline{Q_2})(Q_1)(\overline{Q_0})(cuenta = 25) + (\overline{Q_2})(Q_1)(Q_0)(\overline{cuenta = 30}) + \\
 & +(Q_2)(\overline{Q_1})(\overline{Q_0})(cuenta = 38) + (Q_2)(\overline{Q_1})(Q_0)(\overline{cuenta = 45}) + \\
 & +(Q_2)(Q_1)(\overline{Q_0})(cuenta = 55) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

$$\begin{aligned}
 D_1(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(\overline{Q_1})(Q_0)(cuenta = 17) + (\overline{Q_2})(Q_1)(\overline{Q_0})(cuenta = 25) + \\
 & +(\overline{Q_2})(Q_1)(\overline{Q_0})(\overline{cuenta = 25}) + (\overline{Q_2})(Q_1)(Q_0)(\overline{cuenta = 30}) + \\
 & +(Q_2)(\overline{Q_1})(Q_0)(cuenta = 45) + (Q_2)(Q_1)(\overline{Q_0})(cuenta = 55) + \\
 & +(Q_2)(Q_1)(\overline{Q_0})(\overline{cuenta = 55}) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

$$\begin{aligned}
 D_2(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(Q_1)(Q_0)(cuenta = 30) + (Q_2)(\overline{Q_1})(\overline{Q_0})(\overline{cuenta = 38}) + \\
 & +(Q_2)(\overline{Q_1})(\overline{Q_0})(cuenta = 38) + (Q_2)(\overline{Q_1})(Q_0)(\overline{cuenta = 45}) + \\
 & +(Q_2)(\overline{Q_1})(Q_0)(cuenta = 45) + (Q_2)(Q_1)(\overline{Q_0})(\overline{cuenta = 55}) + \\
 & +(Q_2)(Q_1)(\overline{Q_0})(cuenta = 55) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

Reduciendo por Identidad de la Lógica de Boole referida a $[a + \bar{a} = 1]$:

$$\begin{aligned}
 D_1(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(\overline{Q_1})(Q_0)(cuenta = 17) + (\overline{Q_2})(Q_1)(\overline{Q_0}) + \\
 & +(\overline{Q_2})(Q_1)(Q_0)(\overline{cuenta = 30}) + (Q_2)(\overline{Q_1})(Q_0)(cuenta = 45) + \\
 & +(Q_2)(Q_1)(\overline{Q_0}) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

$$\begin{aligned}
 D_2(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(Q_1)(Q_0)(cuenta = 30) + (Q_2)(\overline{Q_1})(\overline{Q_0}) + \\
 & +(Q_2)(\overline{Q_1})(Q_0) + (Q_2)(Q_1)(\overline{Q_0}) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

6.8.Programa 7

Así, la máquina de estados del Programa 7 consiste en 8 estados, correspondientes al uso de 3 flip-flops de 1 bit o 1 flip flop de 3, representados en forma cíclica según el diagrama de la máquina de estados de la figura siguiente.

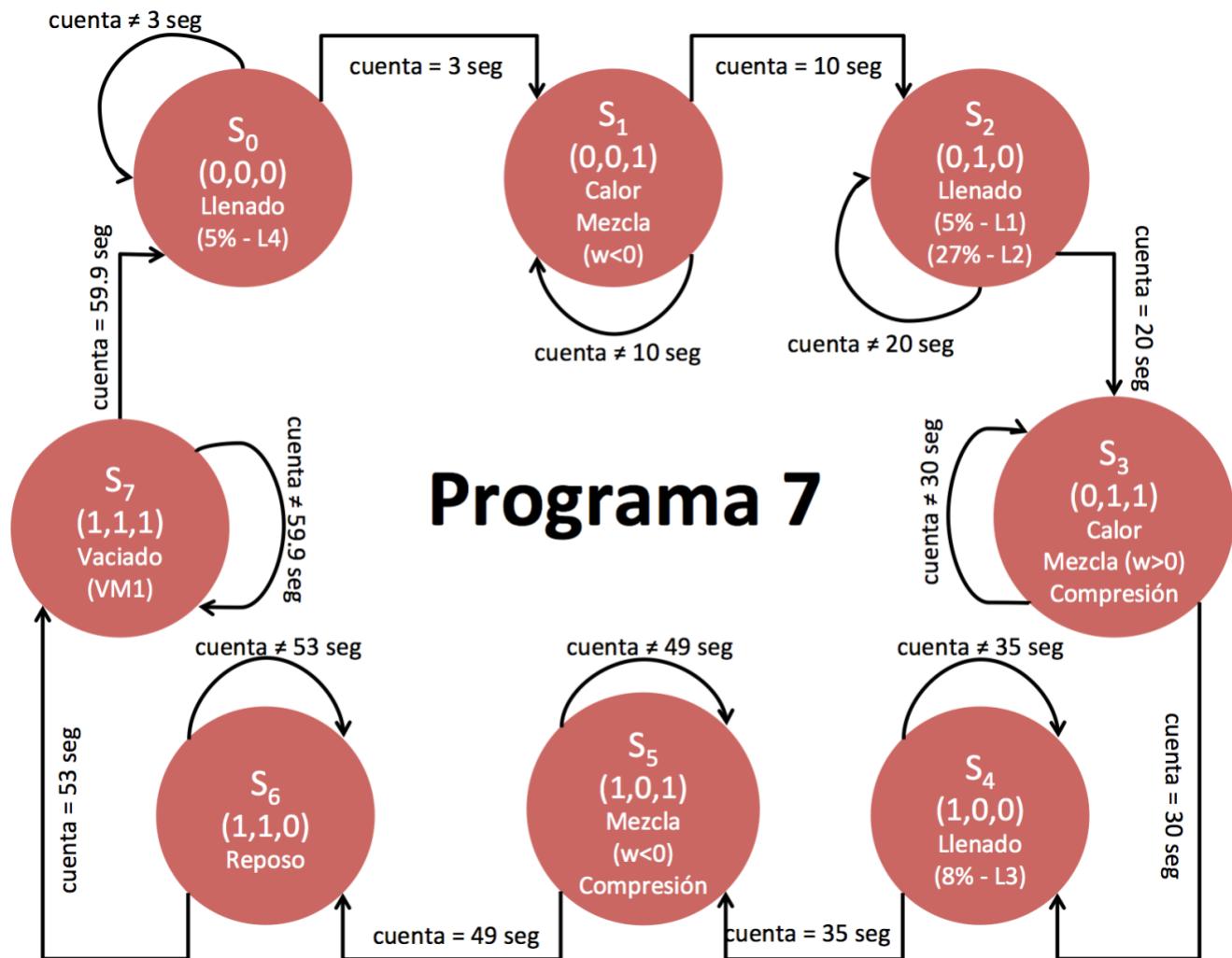


Figura 18: Máquina de Estados del Programa 7

La tabla correspondiente a la máquina de estados anterior se muestra en la tabla adjunta a continuación.

Estado Actual				Entradas de los Flip-Flop tipo D			Condición	Estado Siguiente			
S_i	$Q_2(i)$	$Q_1(i)$	$Q_0(i)$	D_2	D_1	D_0		S_{i+1}	$Q_2(i+1)$	$Q_1(i+1)$	$Q_0(i+1)$
S_0	0	0	0	0	0	0	$cuenta \neq 3$	S_0	0	0	0
S_0	0	0	0	0	0	1	$cuenta = 3$	S_1	0	0	1
S_1	0	0	1	0	0	1	$cuenta \neq 10$	S_1	0	0	1
S_1	0	0	1	0	1	0	$cuenta = 10$	S_2	0	1	0
S_2	0	1	0	0	1	0	$cuenta \neq 20$	S_2	0	1	0
S_2	0	1	0	0	1	1	$cuenta = 20$	S_3	0	1	1
S_3	0	1	1	0	1	1	$cuenta \neq 30$	S_3	0	1	1
S_3	0	1	1	1	0	0	$cuenta = 30$	S_4	1	0	0
S_4	1	0	0	1	0	0	$cuenta \neq 35$	S_4	1	0	0
S_4	1	0	0	1	0	1	$cuenta = 35$	S_5	1	0	1
S_5	1	0	1	1	0	1	$cuenta \neq 49$	S_5	1	0	1
S_5	1	0	1	1	1	0	$cuenta = 49$	S_6	1	1	0
S_6	1	1	0	1	1	0	$cuenta \neq 53$	S_6	1	1	0
S_6	1	1	0	1	1	1	$cuenta = 53$	S_7	1	1	1
S_7	1	1	1	1	1	1	$cuenta \neq 59.9$	S_7	1	1	1
S_7	1	1	1	0	0	0	$cuenta = 59.9$	S_0	0	0	0

Tabla 9. Tabla de Verdad o Lógica referente a la máquina de estado tratada referente al Programa 1

Desarrollo por Suma de Productos:

$$\begin{aligned}
 D_0(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(\overline{Q_1})(\overline{Q_0})(cuenta = 3) + (\overline{Q_2})(\overline{Q_1})(Q_0)(\overline{cuenta = 10}) + \\
 & +(\overline{Q_2})(Q_1)(\overline{Q_0})(cuenta = 20) + (\overline{Q_2})(Q_1)(Q_0)(\overline{cuenta = 30}) + \\
 & +(Q_2)(\overline{Q_1})(\overline{Q_0})(cuenta = 35) + (Q_2)(\overline{Q_1})(Q_0)(\overline{cuenta = 49}) + \\
 & +(Q_2)(Q_1)(\overline{Q_0})(cuenta = 53) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

$$\begin{aligned}
 D_1(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(\overline{Q_1})(Q_0)(cuenta = 10) + (\overline{Q_2})(Q_1)(\overline{Q_0})(cuenta = 20) + \\
 & +(\overline{Q_2})(Q_1)(\overline{Q_0})(\overline{cuenta = 20}) + (\overline{Q_2})(Q_1)(Q_0)(\overline{cuenta = 30}) + \\
 & +(Q_2)(\overline{Q_1})(Q_0)(cuenta = 49) + (Q_2)(Q_1)(\overline{Q_0})(cuenta = 53) + \\
 & +(Q_2)(Q_1)(\overline{Q_0})(\overline{cuenta = 53}) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

$$\begin{aligned}
 D_2(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(Q_1)(Q_0)(cuenta = 30) + (Q_2)(\overline{Q_1})(\overline{Q_0})(\overline{cuenta = 35}) + \\
 & +(Q_2)(\overline{Q_1})(\overline{Q_0})(cuenta = 35) + (Q_2)(\overline{Q_1})(Q_0)(\overline{cuenta = 49}) + \\
 & +(Q_2)(\overline{Q_1})(Q_0)(cuenta = 49) + (Q_2)(Q_1)(\overline{Q_0})(\overline{cuenta = 53}) + \\
 & +(Q_2)(Q_1)(\overline{Q_0})(cuenta = 53) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

Reduciendo por Identidad de la Lógica de Boole referida a $[a + \bar{a} = 1]$:

$$\begin{aligned}
 D_1(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(\overline{Q_1})(Q_0)(cuenta = 10) + (\overline{Q_2})(Q_1)(\overline{Q_0}) + \\
 & +(\overline{Q_2})(Q_1)(Q_0)(\overline{cuenta = 30}) + (Q_2)(\overline{Q_1})(Q_0)(cuenta = 49) + \\
 & +(Q_2)(Q_1)(\overline{Q_0}) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

$$\begin{aligned}
 D_2(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(Q_1)(Q_0)(cuenta = 30) + (Q_2)(\overline{Q_1})(\overline{Q_0}) + \\
 & +(Q_2)(\overline{Q_1})(Q_0) + (Q_2)(Q_1)(\overline{Q_0}) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

6.9. Programa 8

De la misma manera, la máquina de estados del Programa 8 consiste en 8 estados, correspondientes al uso de 3 flip-flops de 1 bit o 1 flip flop de 3, representados en forma cíclica según el diagrama de la máquina de estados de la figura siguiente.

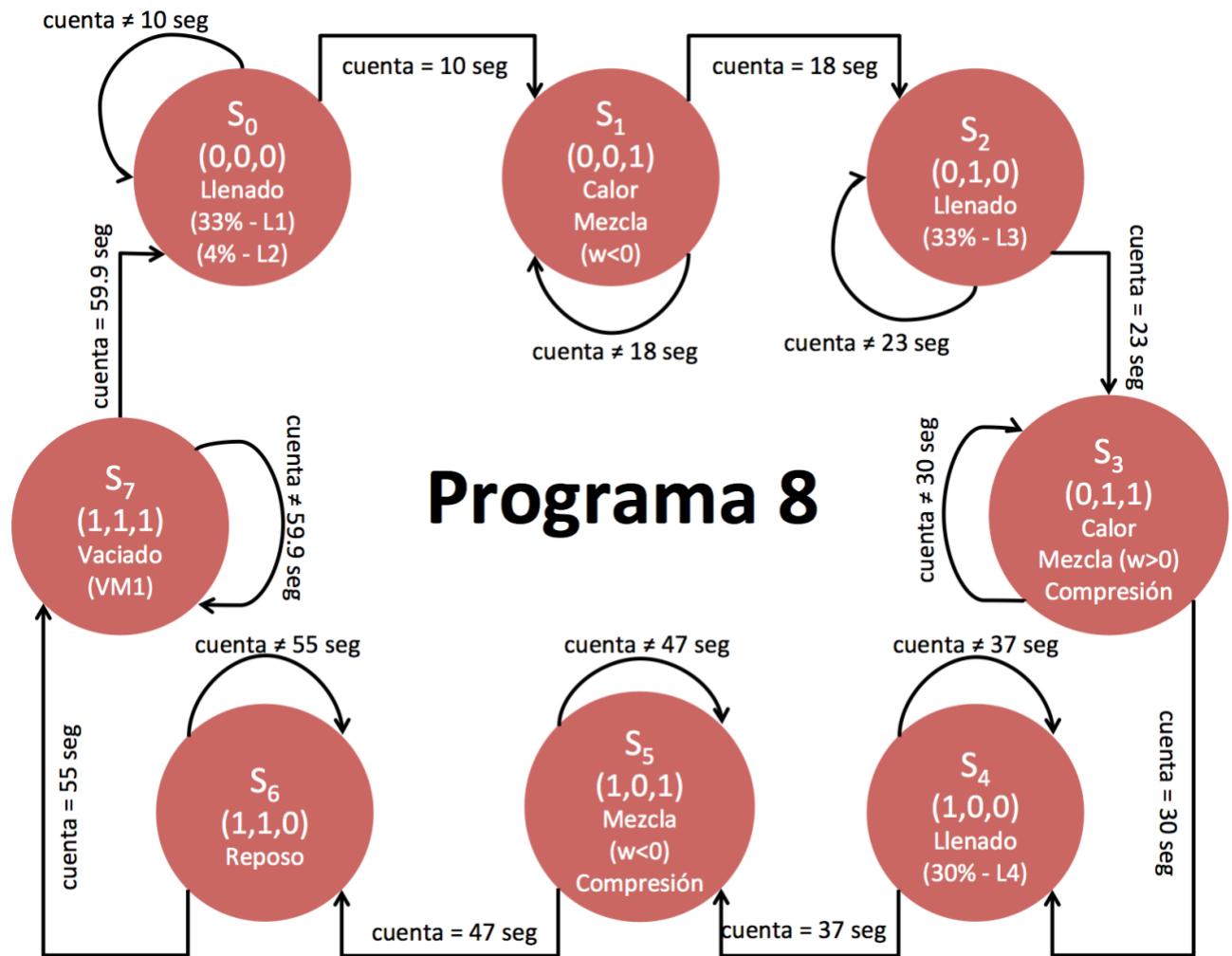


Figura 19: Máquina de Estados del Programa 8

La tabla correspondiente a la máquina de estados anterior se muestra en la tabla adjunta a continuación.

Estado Actual				Entradas de los Flip-Flop tipo D			Condición	Estado Siguiente			
S_i	$Q_2(i)$	$Q_1(i)$	$Q_0(i)$	D_2	D_1	D_0		S_{i+1}	$Q_2(i+1)$	$Q_1(i+1)$	$Q_0(i+1)$
S_0	0	0	0	0	0	0	$cuenta \neq 10$	S_0	0	0	0
S_0	0	0	0	0	0	1	$cuenta = 10$	S_1	0	0	1
S_1	0	0	1	0	0	1	$cuenta \neq 18$	S_1	0	0	1
S_1	0	0	1	0	1	0	$cuenta = 18$	S_2	0	1	0
S_2	0	1	0	0	1	0	$cuenta \neq 23$	S_2	0	1	0
S_2	0	1	0	0	1	1	$cuenta = 23$	S_3	0	1	1
S_3	0	1	1	0	1	1	$cuenta \neq 30$	S_3	0	1	1
S_3	0	1	1	1	0	0	$cuenta = 30$	S_4	1	0	0
S_4	1	0	0	1	0	0	$cuenta \neq 37$	S_4	1	0	0
S_4	1	0	0	1	0	1	$cuenta = 37$	S_5	1	0	1
S_5	1	0	1	1	0	1	$cuenta \neq 47$	S_5	1	0	1
S_5	1	0	1	1	1	0	$cuenta = 47$	S_6	1	1	0
S_6	1	1	0	1	1	0	$cuenta \neq 55$	S_6	1	1	0
S_6	1	1	0	1	1	1	$cuenta = 55$	S_7	1	1	1
S_7	1	1	1	1	1	1	$cuenta \neq 59.9$	S_7	1	1	1
S_7	1	1	1	0	0	0	$cuenta = 59.9$	S_0	0	0	0

Tabla 10. Tabla de Verdad o Lógica referente a la máquina de estado tratada referente al Programa 8

Desarrollo por Suma de Productos:

$$\begin{aligned}
 D_0(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(\overline{Q_1})(\overline{Q_0})(cuenta = 10) + (\overline{Q_2})(\overline{Q_1})(Q_0)(\overline{cuenta = 18}) + \\
 & +(\overline{Q_2})(Q_1)(\overline{Q_0})(cuenta = 23) + (\overline{Q_2})(Q_1)(Q_0)(\overline{cuenta = 30}) + \\
 & +(Q_2)(\overline{Q_1})(\overline{Q_0})(cuenta = 37) + (Q_2)(\overline{Q_1})(Q_0)(\overline{cuenta = 47}) + \\
 & +(Q_2)(Q_1)(\overline{Q_0})(cuenta = 55) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

$$\begin{aligned}
 D_1(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(\overline{Q_1})(Q_0)(cuenta = 18) + (\overline{Q_2})(Q_1)(\overline{Q_0})(cuenta = 23) + \\
 & +(\overline{Q_2})(Q_1)(\overline{Q_0})(\overline{cuenta = 23}) + (\overline{Q_2})(Q_1)(Q_0)(\overline{cuenta = 30}) + \\
 & +(Q_2)(\overline{Q_1})(Q_0)(cuenta = 47) + (Q_2)(Q_1)(\overline{Q_0})(cuenta = 55) + \\
 & +(Q_2)(Q_1)(\overline{Q_0})(\overline{cuenta = 55}) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

$$\begin{aligned}
 D_2(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(Q_1)(Q_0)(cuenta = 30) + (Q_2)(\overline{Q_1})(\overline{Q_0})(\overline{cuenta = 37}) + \\
 & +(Q_2)(\overline{Q_1})(\overline{Q_0})(cuenta = 37) + (Q_2)(\overline{Q_1})(Q_0)(\overline{cuenta = 47}) + \\
 & +(Q_2)(\overline{Q_1})(Q_0)(cuenta = 47) + (Q_2)(Q_1)(\overline{Q_0})(\overline{cuenta = 55}) + \\
 & +(Q_2)(Q_1)(\overline{Q_0})(cuenta = 55) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

Reduciendo por Identidad de la Lógica de Boole referida a $[a + \bar{a} = 1]$:

$$\begin{aligned}
 D_1(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(\overline{Q_1})(Q_0)(cuenta = 18) + (\overline{Q_2})(Q_1)(\overline{Q_0}) + \\
 & +(\overline{Q_2})(Q_1)(Q_0)(\overline{cuenta = 30}) + (Q_2)(\overline{Q_1})(Q_0)(cuenta = 47) + \\
 & +(Q_2)(Q_1)(\overline{Q_0}) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

$$\begin{aligned}
 D_2(Q_2, Q_1, Q_0, cuenta) = & (\overline{Q_2})(Q_1)(Q_0)(cuenta = 30) + (Q_2)(\overline{Q_1})(\overline{Q_0}) + \\
 & +(Q_2)(\overline{Q_1})(Q_0) + (Q_2)(Q_1)(\overline{Q_0}) + (Q_2)(Q_1)(Q_0)(\overline{cuenta = 59.9})
 \end{aligned}$$

6.10. Compilación de la Lógica de los Programas

La programación de los programas consistió en la debida descripción de las máquinas de estados anteriores, debiéndose así usar o crear componentes vistos en el curso tales como flip-flop, decodificadores o multiplexores y compuertas lógicas, representados por el diagrama de bloques siguiente para el Programa 1.

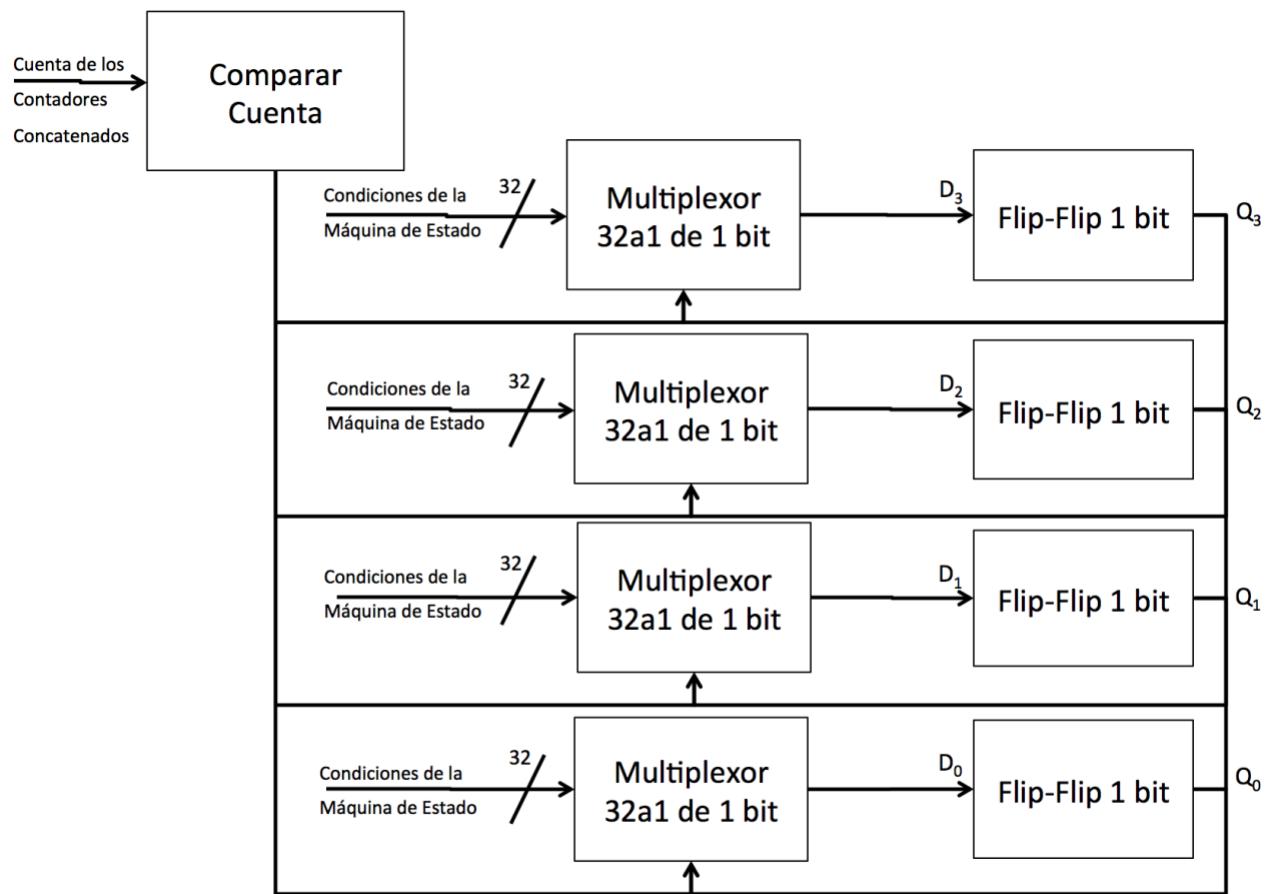


Figura 20. Diagrama de bloques del programa VHDL referido al Programa 1

Y un diagrama de bloques análogo para el caso de los restantes programas, como se muestra en la imagen a continuación. En el Programa 1, al tratarse del uso de 9 estados y por tanto de 4 flip-flop y la consiguiente utilización de 5 bits de control en los multiplexores, el mismo deberá poseer 32 entradas (2^5 inputs). Mientras, en los siguientes programas se decidieron 8 estados, consecuentemente 3 flip-flop y así el uso de 4 bits de control en los multiplexores, los cuales poseerán de esta forma 16 entradas (2^4 inputs).

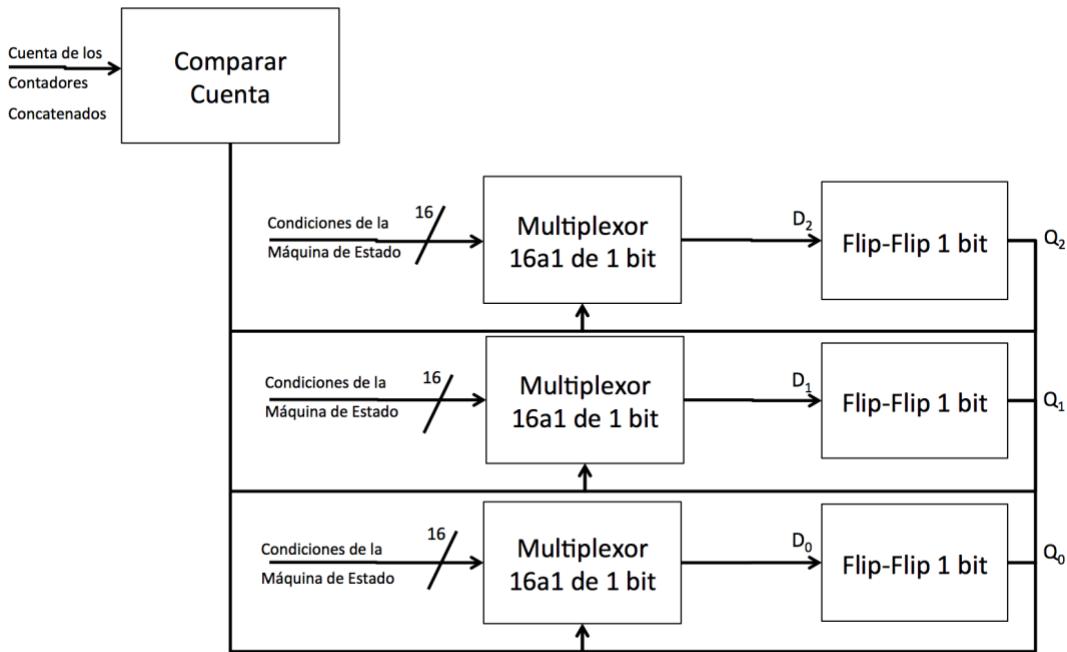


Figura 21. Diagrama de bloques del programa VHDL referido a los Programas 2 a 8

El bloque de CompararCuenta, encargado de comparar tres vectores de 4 bits consistentes en las correspondientes décimas, unidades y decenas de segundo recibidas de los tres contadores conectados en cascada y teniendo de entrada el reloj de 10 Hz (0.1 segundos de período), determina cuando la cuenta referida es igual (o diferente) al valor ingresado como valor a comparar para cada cifra del número. Así, el valor lógico de la salida será 1 si y solo si en las tres comparaciones realizadas dentro del bloque el valor lógico es 1 (cuenta1 and cuenta2 and cuenta 3). Nótese que esta misma comparación se pudo haber realizado comparando un vector único de 10 bits con la cuenta total (cuyo valor máximo sería el de 2^{10} igual a 1024, precisando únicamente hasta el 599), prefiriéndose realizarlo de la manera anterior al considerarla más cómoda de usar y prolífica en su accionar.

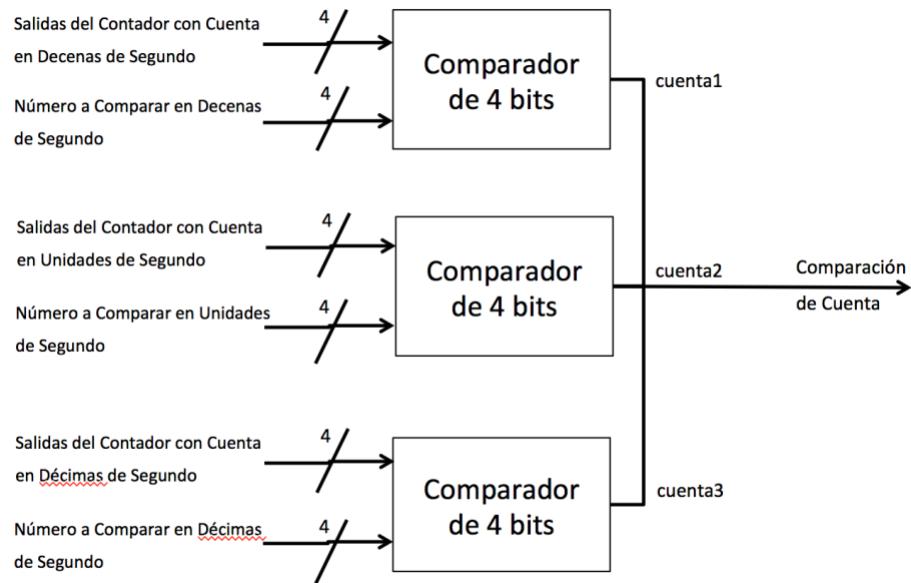


Figura 22. Diagrama de bloques del comparador excepcional creado para comparar cifras del tiempo transcurrido

Finalmente, cada estado determina la acción realizada por la/s mezcladora/s, determinado por tanto por el valor lógico de las cuatro (Programa 1) o tres funciones lógicas de las entradas de los flip-flops. La implementación de las acciones mencionadas se deben mostrar en el dispositivo, por lo que se precisa que a partir del estado en el que el programa se encuentre se despliegue en los correspondientes *displays* de 7 segmentos y LEDs de la FPGA. De esta manera, se debieron crear salidas del VHDL de cada programa que indiquen si se encontraba en el estado de llenado (y de cual válvula), compresión, calentamiento, vaciado o en proceso de mezcla (diferenciando entre horaria y antihoraria).

Para indicar el estado del llenado, se utilizó un vector de 4 bits referido a las cuatro respectivas válvulas para de esta forma determinar cuando y cual se encuentra abierta en el estado actual del programa. De la misma manera, para la compresión, calentamiento y vaciado se precisó una señal independiente para cada uno (de forma de mantener la prolijidad del código), la cual posea valor lógico 0 si no se está usando o 1 si se encuentra en algún estado que se esté utilizando actualmente. Para el proceso de mezcla, y con el objetivo de diferenciar el sentido, se utilizó un vector de 2 bits en el cual si la señal fuera (0,0) se correspondiera con que el contenido de la mezcladora no se encuentra siendo mezclada, (0,1) con que el mismo se esté mezclando en sentido horario, (0,1) en caso de

estarse mezclando en sentido antihorario y (1,1) en caso excepcional, no utilizado, significando lo mismo que si la señal fuera (0,0) (la idea de la representación por dos bits está dada en la letra del obligatorio). Estos valores enviados por las máquinas de estados de cada programa se capturan por el Macro Programa explicado y analizado posteriormente en el documento, los cuales se envían a la Macro Pantalla para su correspondiente y coherente muestra en el dispositivo según las regulaciones especificadas por el Obligatorio.

Nota Importante: Como se puede observar, la lógica y despliegue de los programas ha concluido en un proceso cíclico, es decir, una vez que se inicia un programa se repite hasta que no se le asigne otro, continuando en tal caso el siguiente programa a partir de la cuenta del contador decimal a la que el anterior programa fue cambiado. Esta fue meramente una decisión de diseño del equipo, puesto que en la letra del Obligatorio 2 no especificaba nada al respecto. Esta decisión se basó en una consideración de practicidad y simplicidad para el uso de menos cableado. Asimismo, para este cometido, en los programas con instrucciones hasta antes de los 60 segundos (por ejemplo, el programa) se los dejó en reposo en los siguientes segundos hasta completar el minuto. Sin embargo, conózcase que, en caso de tener que haber procesado un programa con un comienzo y fin y de duración 60 segundos (terminando al final de cada segundo), se pudo haber realizado con tal solo agregarle una condición concatenada con una función lógica OR al *Carry In* del contador decimal, o con el hecho de agregarle un nuevo estado de uso a las máquinas de estado de todos los programas, al cual entre al terminar el proceso y cuya condición de permanencia sea mientras no se configure otro programa o se lo introduzca manualmente. De aquí la cuestión de comodidad, puesto que en el caso de los programas 2 a 8, al agregarse un nuevo estado se hubiese tenido que agregar un nuevo flip-flop. Se reitera igualmente que este funcionamiento se pudo haber implementado sencillamente como es descrito en caso de haber sido necesario.

7. Despliegue de Pantallas Requeridas

7.1. Convertidor de Bits a BCD

A la hora de mostrar la velocidad o temperatura en los *displays* de 7 segmentos, al ser un número de 8 bits que puede tomar un valor de hasta “255”, este no se puede desplegar en la pantalla mediante un simple decodificador de 7 segmentos, por lo que se requirió de este conversor.

La idea, de este modo, es separar el número de 8 bits en 3 de 4 bits y así decodificarlos individualmente a cada *display*. El principio de funcionamiento del circuito elegido está dado por un algoritmo conocido como “desplazar a la izquierda y sumar tres” que consta de desplazar el número binario hacia la izquierda un lugar hasta que este número obtenido a la izquierda de número inicial (en este caso de 8 bits) obtenga un valor mayor o igual a 5, en tal caso se debe sumar 3 a este número y se continúa el proceso desplazando el número hacia la izquierda y sumando 3 si se cumple la condición hasta finalmente obtener los 3 números de 4 bits. Habiéndose ya contemplado la variedad de soluciones disponibles en internet y que todas ellas eran mediante un proceso (“process”) (herramienta disponible en VHDL), finalmente se optó por crear una versión en diseño de hardware que realizara la conversión. Para dicha solución se necesitó la creación de un componente que sumase 3 si el número de entrada es mayor o igual a 5, se pudo realizar mediante un sumador de 4 bits que sumase 3 y un comparador que comparase la cuenta de entrada con 5. Finalmente para la realización del convertidor de binario a BCD, se instanciaron los necesarios sumadores y se dispusieron las conexiones entre ellos tal como indica la imagen anterior.

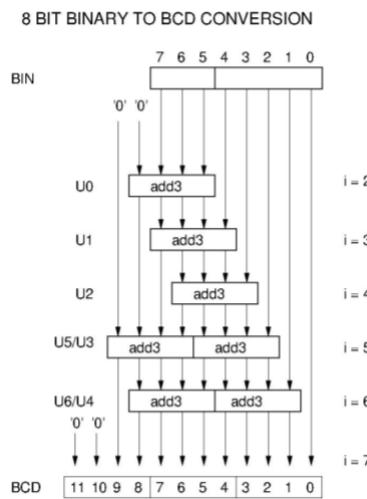


Figura 23. Diagrama de bloques de un Convertidor de bits a BCD a modo de ejemplo respecto al realizado

7.2. Pantalla 1

Se necesitó que al seleccionar con los *switchs* pertinentes se despliegue la información predefinida por el formato de la Pantalla 1 ($SW8=0$ y $SW9=0$) , se debía desplegar en HEX3 (*display* de 7 segmentos del extremo izquierdo) una animación del proceso actual, analizadas posteriormente en el documento, mientras que en los demás *displays* (HEX2, HEX1, HEX0) se debía mostrar el transcurso del tiempo en decenas, unidades y décimas de segundo.

De esta forma, se utilizó el contador decimal previamente explicitado, relacionado a los pines correspondientes a los tres *displays* necesitados. Por otro lado, para mostrar apropiada y coherentemente la animación respecto al tiempo transcurrido y a la acción respectiva, se prosiguió a implementar un multiplexor de 8 bits (1 byte) de 4 entradas (las cuatro animaciones posibles) y 1 salida donde los 2 bits de control necesitados fueran los dos *switchs* que determinan cual pantalla se quiere desplegar. Así, se vio necesario que el programa en VHDL reciba como entradas las señales lógicas correspondientes para que el mismo recoja y despliegue acordeamente cual acción se está llevando a cabo en cada momento. Esta información llega a la pantalla desde el programa correspondiente al que está aconteciendo en el instante actual, el cual luego de determinarlo es enviado al bloque del Macro Programa y de allí al bloque de la Macro Pantalla, el cual se encarga de enviar a cada programa VHDL de las pantallas la información necesaria para su correcto funcionamiento.

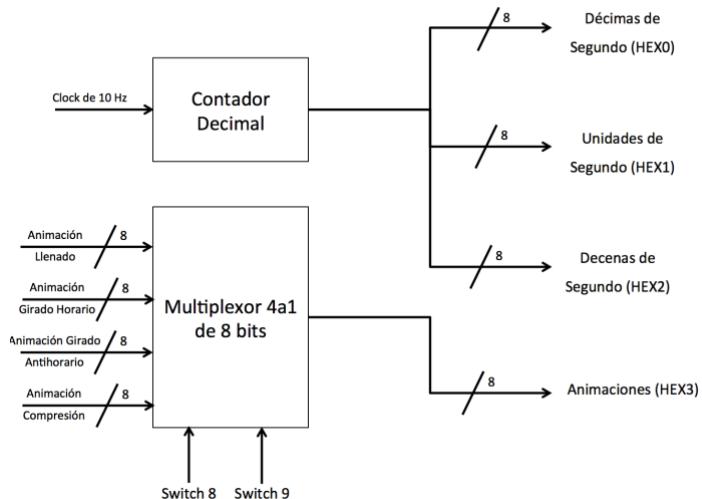


Figura 24. Diagrama de bloques de los componentes instanciados dentro de la Pantalla 1 para su correcto funcionamiento

7.3. Pantalla 2

Según el formato de la Pantalla 2, se precisó que en caso de que los *switchs* se codificaran de forma de tener que desplegar la información debida ($SW8=0$ y $SW9=1$), se mostrara en el *display* de 7 segmentos del extremo izquierdo (HEX3) el signo de la velocidad de girado o mezclado, siendo este negativo en caso de ser antihorario y positivo en caso contrario. Asimismo, en los demás *displays* (HEX2, HEX1, HEX0) se debió mostrar la correspondiente velocidad configurada en la mezcladora, ingresada por los bytes de comando y datos previamente mencionados.

Para realizar esta operación, se hizo uso de multiplexores y el recientemente creado y analizado convertidor de bits a BCD. Al recibir el programa VHDL de la Pantalla 2 el vector de 2 bits correspondiente a la activación del proceso de rotar (o mezclar) el contenido de la mezcladora junto con el sentido con el que lo hace, se continuó a utilizar un multiplexor de 2 bits de 4 entradas y 1 salida, donde los 2 bits de control fueran los respectivos elementos del vector mencionado. Este vector, de esta forma, poseerá valor lógico (0,0) en caso de no estar mezclando el contenido del recipiente, mientras que si tiene valor lógico (0,1) está mezclando en sentido horario, (1,0) si lo hace en sentido antihorario y (1,1) si no está mezclando. Así, en caso de poseer el vector mencionado un valor lógico (0,1), el *display* no deberá mostrar nada, mientras que si tiene un valor lógico (1,0) deberá enseñar un signo de menos. De la misma manera, al recibir el valor de la velocidad de la mezcla por el byte de comandos y datos, esta tendrá formato de 8 bits (1 byte) y por tanto se encontrará limitada por un valor máximo de 225 (máximo resultado posible de la multiplicación de 4 bits), necesitando convertirla en tres señales separadas de 4 bits de forma de decodificar cada una en un decodificador de 7 segmentos y mostrarla en los correspondientes *displays* de 7 segmentos. Para este último proceso se precisó necesariamente la inclusión del bloque encargado de convertir bits a BCD analizado páginas atrás, mostrando el número en caso que las señales lógicas brindadas por los *switchs* 8 y 9 fueran diferentes de (0,0) y (1,1).

Nótese que el despliegue de los valores de la velocidad como de la temperatura es un proceso instantáneo y por tanto ninguno de los valores mencionados poseen características incrementales, de forma de simplificar la muestra de información. Sin embargo, esta nueva implementación se pudo haber realizado al calcular el incremento necesario para la velocidad y la temperatura para cada

instante de tiempo considerado (en este caso, décimas de segundo) o colocando un sensor apropiado a cada mezcladora.

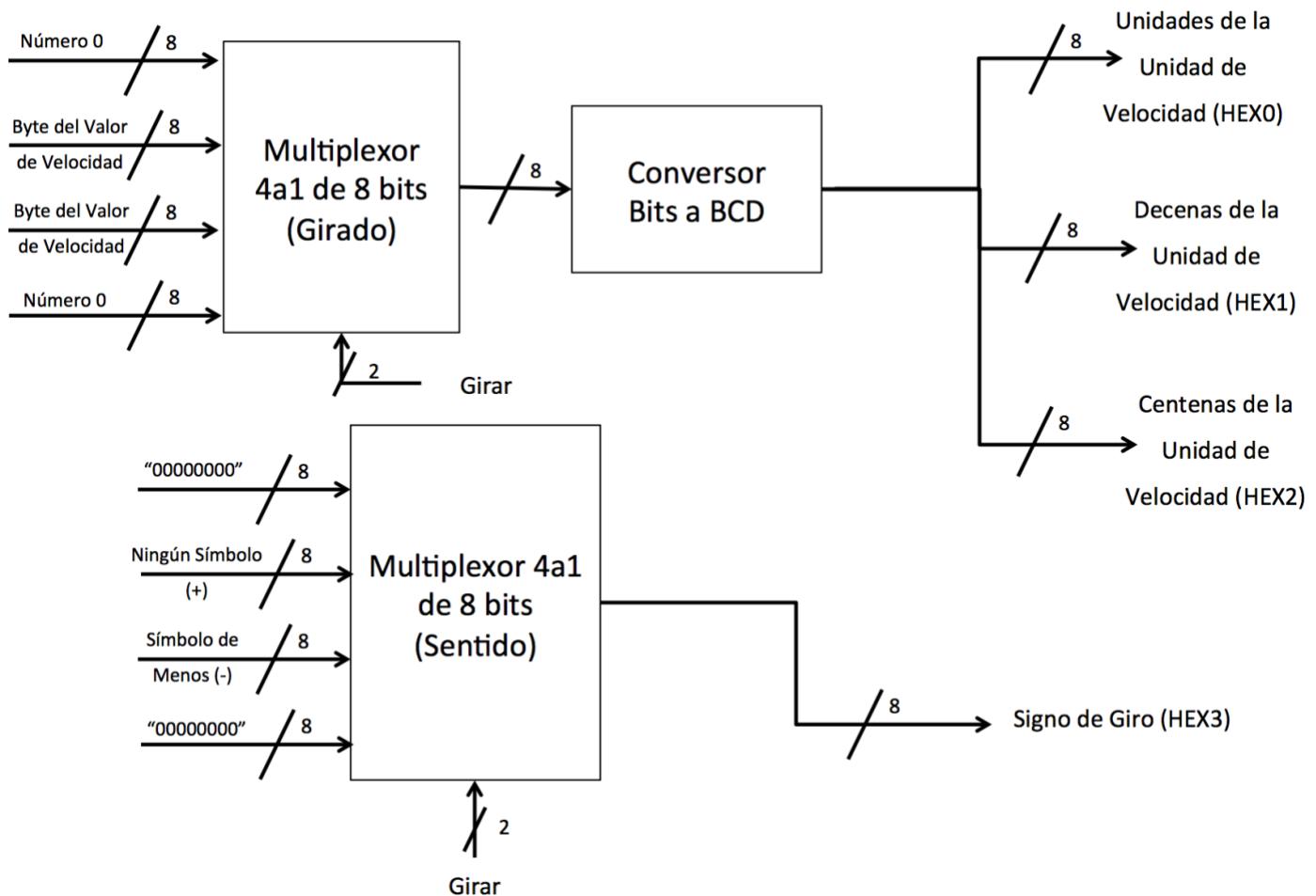


Figura 25. Diagrama de bloques referido a los componentes internos a la Pantalla 2 para su adecuado labor

7.4. Pantalla 3

Continuando con el formato de la Pantalla 3, desplegado cuando los *switchs* se codifiquen de manera de tener que desplegar la información debida ($SW8=1$ y $SW9=0$), se mostrará en los *displays* de 7 segmentos correspondientes a HEX2, HEX1 y HEX0 (es decir, todos menos el *display* del extremo izquierdo) el valor de la temperatura indicada para la mezcla indicada previamente por los bytes de comando y datos y transmitidos por el RS 232. El *display* del extremo izquierdo (HEX3) se dejó apagado al no indicarse en la letra del trabajo ninguna condición que el mismo deba cumplir.

Análogamente a lo estudiado y realizado para el caso de la Pantalla 2, se prosiguió a usar un bloque del conversor de bits a BCD, al cual entra el byte (los 8 bits) transmitidos por el 232 a la placa FPGA y posee como salidas tres vectores de 4 bits en los cuales se codifica las centenas, decenas y unidades del valor enviado para la temperatura. Igual que en el caso anterior, si en el estado del programa actual no se encuentra suministrando calor al sistema de las mezcladoras por medio de la resistencia teórica, se desplego como decisión de diseño el número 0 en los tres *displays* de modo de visualizar claramente que no se está aportando energía térmica. Para ello, en este momento se prosiguió a utilizar un multiplexor de 8 bits de 2 entradas y 1 salida con el bit de control correspondiente al valor recibido de entrada por el programa de VHDL de Pantalla 3 respectivo a la señal lógica enviada por el programa actualmente en acción que indique con valor lógico 1 si efectivamente está ingresando calor al sistema o 0 en caso contrario.

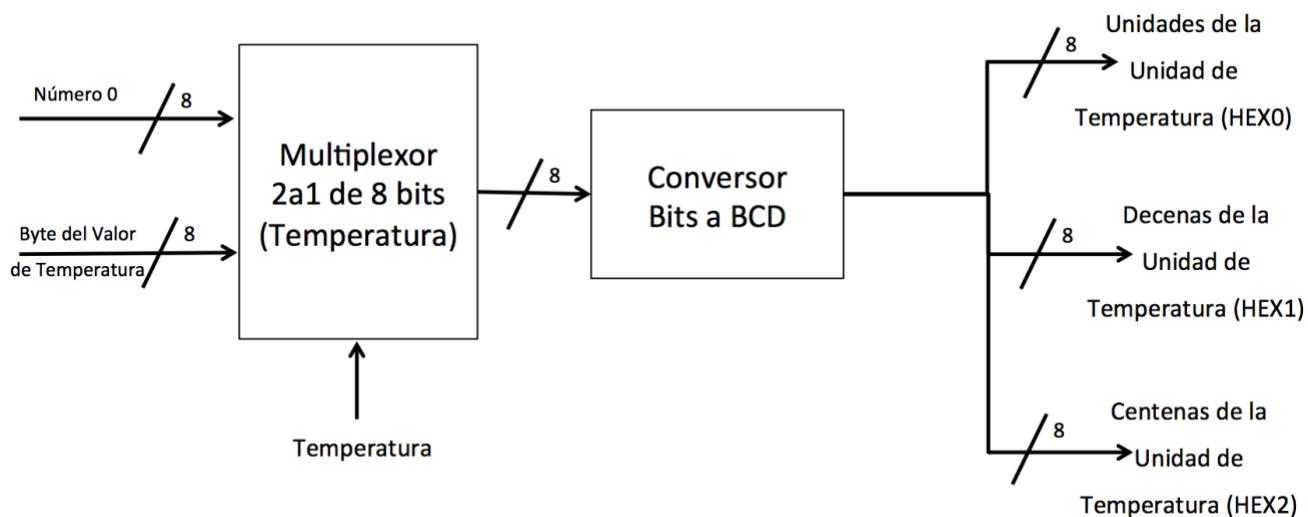


Figura 26. Diagrama de bloques de la Pantalla 3 indicando los cableados internos que justifican su accionar

7.5. Pantalla 4

Para la Pantalla 4, activada al codificarse apropiadamente los *switchs* correspondientes ($SW8 = 1$ y $SW9 = 1$), se solicitó que tenga un formato consistente en indicar el porcentaje de líquido que llena cada una de las cuatro válvulas de entrada de cada mezcladora coherentemente con el programa actual configurado. Así, el número mencionado debió ser desplegado en los dos *displays* de 7 segmentos de la derecha (HEX1 y HEX0), mientras que en el siguiente *display* (HEX2) debía mostrar de cual válvula se trataba de la mezcladora presentemente visualizada junto a la muestra de la letra “L” en el *display* restante correspondiente al del extremo izquierdo (HEX3). La información de cada una de las 4 válvulas de la mezcladora seleccionada se debían alternar cada 2 segundos, comenzando por el porcentaje de la válvula LM1 hasta la LM4 y volviendo a repetir el ciclo.

Para la realización de esta parte del Obligatorio, a efectos de poseer una mayor comodidad y mantener la prolijidad al programar, se recurrió a generar un modulador de 0.5 Hz (2 segundos de período) analizado posteriormente de tal modo que sea útil al momento de utilizar un contador para conservar la cuenta necesaria para la máquina de estados a realizar. La misma debió consistir de 4 estados respectivos a las 4 válvulas posibles a accionar en cada programa y por consiguiente se debieron hacer uso de 2 flip-flop de 1 bit (existiendo la posibilidad de también haber usado un flip-flop de 2 bits). A su vez, la mencionada máquina de estados debía contener un multiplexor de 8 entradas y 1 salida para cada flip-flop de 1 bit, donde los 3 bits de control consistieran en las salidas de los 2 flip-flop mencionados y la cuenta del contador conectado al modulador de 0.5 Hz.

Estado Actual			Entradas de los Flip-Flop tipo D		Condición	Estado Siguiente		
S_i	$Q_1(i)$	$Q_0(i)$	D_1	D_0		S_{i+1}	$Q_1(i + 1)$	$Q_0(i + 1)$
S_0	0	0	0	0	$cuenta \neq 1$	S_0	0	0
S_0	0	0	0	1	$cuenta = 1$	S_1	0	1
S_1	0	1	0	1	$cuenta \neq 2$	S_1	0	1
S_1	0	1	1	0	$cuenta = 2$	S_2	1	0
S_2	1	0	1	0	$cuenta \neq 3$	S_2	1	0
S_2	1	0	1	1	$cuenta = 3$	S_3	1	1
S_3	1	1	1	1	$cuenta \neq 4$	S_3	1	1
S_3	1	1	0	0	$cuenta = 4$	S_0	0	0

Tabla 11. Tabla de Verdad o Lógica referida a la máquina de estado de la Pantalla 4

Desarrollo por Suma de Productos:

$$D_0(Q_1, Q_0, cuenta) = (\overline{Q_1})(\overline{Q_0})(cuenta = 1) + (\overline{Q_1})(Q_0)(\overline{cuenta = 2}) + \\ + (Q_1)(\overline{Q_0})(cuenta = 3) + (Q_1)(Q_0)(\overline{cuenta = 4})$$

$$D_1(Q_1, Q_0, cuenta) = (\overline{Q_1})(Q_0)(cuenta = 2) + (Q_1)(\overline{Q_0})(cuenta = 3) + \\ + (Q_1)(\overline{Q_0})(\overline{cuenta = 3}) + (Q_1)(Q_0)(\overline{cuenta = 4}) +$$

Reduciendo por Identidad de la Lógica de Boole referida a $[a + \bar{a} = 1]$:

$$D_1(Q_1, Q_0, cuenta) = (\overline{Q_1})(Q_0)(cuenta = 2) + (Q_1)(\overline{Q_0}) + (Q_1)(Q_0)(\overline{cuenta = 4})$$

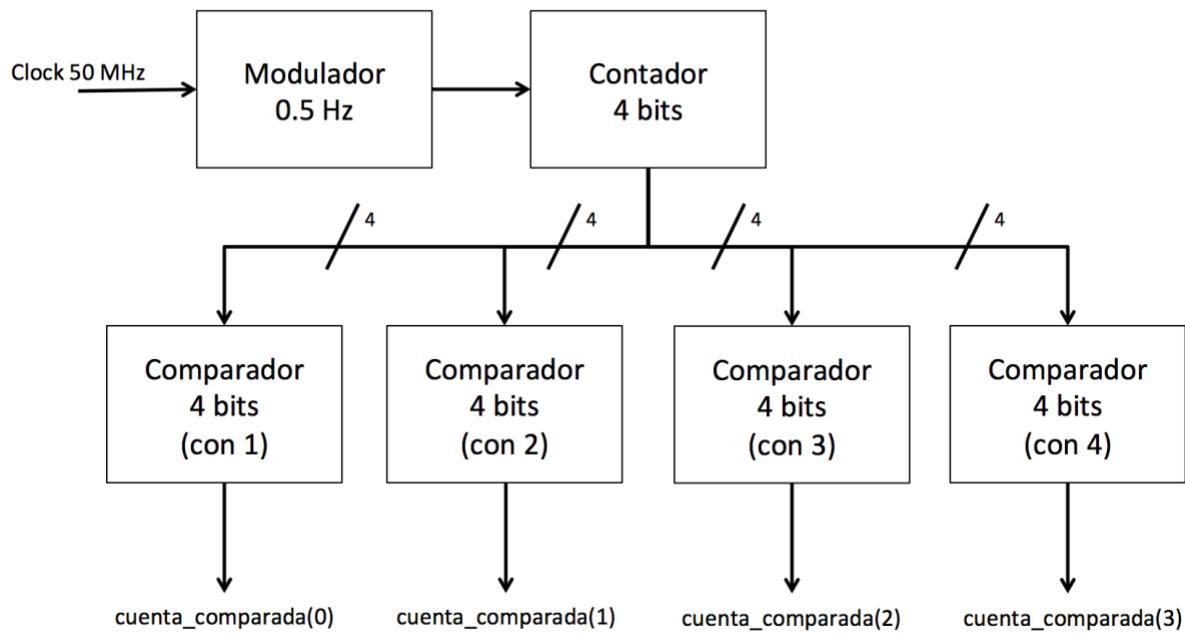


Figura 27. Diagrama de bloques de una sección de la Pantalla 4 que indica el modo de comparar la cuenta recibida por el contador y que produce la caracterización de cada estado

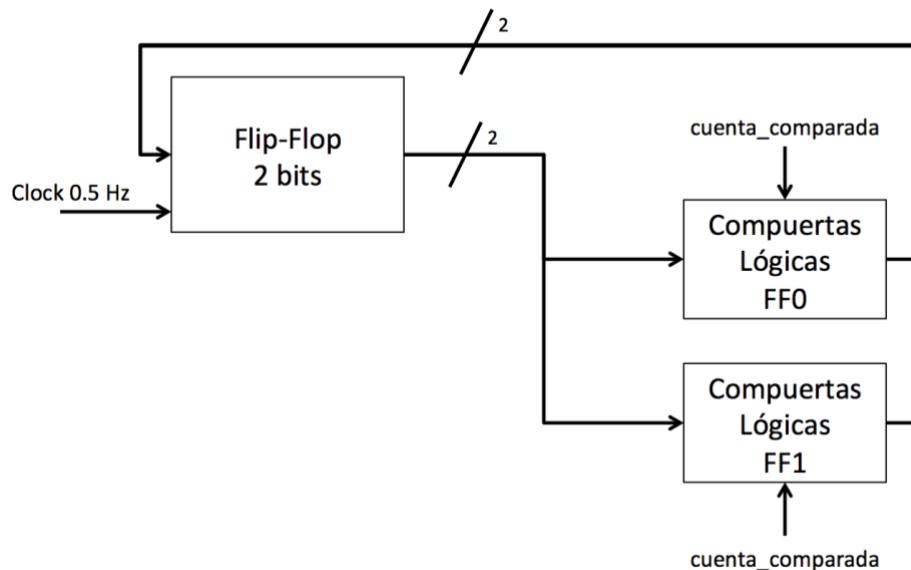


Figura 28. Diagrama de bloques de una sección de la Pantalla 4 que muestra el centro de la máquina de estado anteriormente explicada, variando los estados gracias a las características del Flip-Flop y las relaciones lógicas halladas.

A continuación y para cada *display*, se hicieron uso de multiplexores de 4 entradas y 1 salida de 4 bits, guiados por los 2 bits de control conectados a las entradas de los flip-flops, los cuales en conjunto se encargaran de que símbolo mostrar en cada *display* dependiendo del estado en que se encontrará de la maquia de estado. Es lógico que la salida de los multiplexores mencionados de 4 bits deberán ser decodificados para su correcta muestra en pantalla. El *display* del extremo izquierdo (HEX3) es el único que conserva la misma codificación desde el instante 1, debiendo mostrar la letra “L” independientemente de la máquina de estados. Las entradas de estos últimos multiplexores se corresponden con el valor que se desea mostrar en pantalla para el cada valor que tomen las señales de entrada de los flip-flop, determinantes del estado presente de la máquina de estado central de la Pantalla 4. Para ello, se prosiguió a crear 2 multiplexores de 16 entradas y 1 salida de 4 bits para cada válvula, cuyos 4 bits de control sean aquellos brindados por la entrada del programa VHDL de la Pantalla 4 correspondientes a un vector de 4 bits indicando que programa se encuentra corriendo, puesto que cada válvula solo adquiere un valor (de 2 cifras, de ahí el uso de 2 multiplexores de 4 bits para cada válvula) diferente e independiente por programa. De esta manera y como es indicado más adelante, las salidas de los mencionados 8 multiplexores de 16 a 1 de 4 bits se colocaron en las respectivas entradas de los anteriores multiplexores 4 a 1 de 4 bits encargados del despliegue de información en las pantallas según el formato estipulado y regulado por la Pantalla 4.

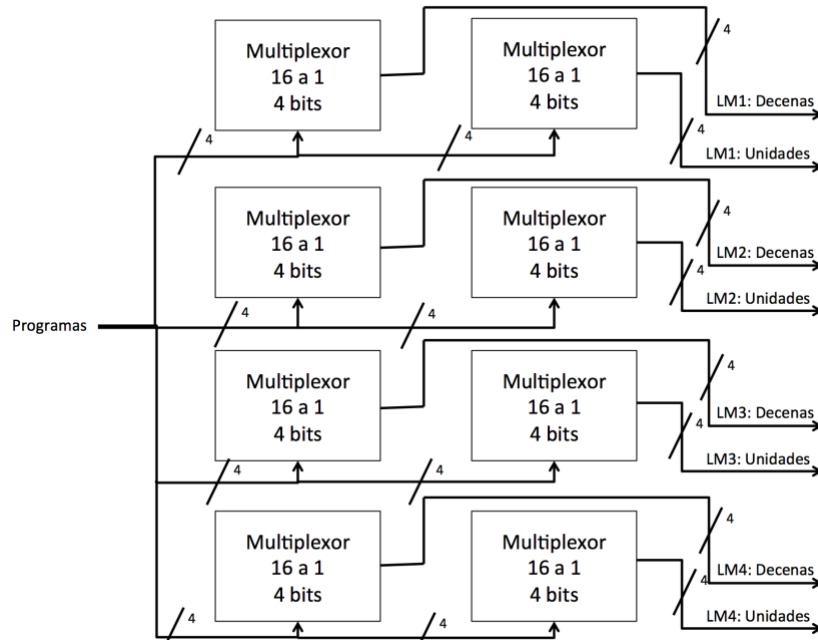


Figura 29. Diagrama de bloques de una sección de la Pantalla 4 que explicita la forma en que los porcentajes de llenado por las válvulas es determinado

8. Moduladores de la Señal XTAL de 50 MHz de la placa DE0 FPGA

8.1. Concepto de Modulador o Divisor de Frecuencia

Comúnmente, se conoce como modulador (o, en el caso del utilizado, como divisor de frecuencia) a un dispositivo electrónico que divide la frecuencia de entrada en una relación casi siempre entera o racional. La forma de la señal de salida puede ser simétrica o asimétrica. La señal de entrada frecuentemente tiene forma de una onda cuadrada pero también puede ser sinusoidal o de otras formas. Suelen estar formados por contadores digitales. Se pueden obtener relaciones de frecuencia no enteras utilizando contadores de módulo variable, por ejemplo, si a cada pulso de salida se cambia el módulo del contador entre 2 y 3, se obtiene una relación de frecuencias de 5:2.

De esta forma, se dice divisor de frecuencia un circuito que recibe en entrada una señal de una frecuencia (determinada f) y da una señal de salida de frecuencia f/n donde n es un número entero. La necesidad de un divisor de frecuencia, ya que tiene tanto con una y la misma señal de reloj debe conducir circuitos en diferentes frecuencias, y porque es más fácil para estabilizar por medio de un circuito en el cuarzo un circuito dado a una tasa superior y luego obtener una frecuencia más baja, que también se ha estabilizado, aunque no es un cristal de cuarzo a la frecuencia deseada.

Así, el objetivo es reducir la frecuencia de entrada, el cual se implementa con ayuda del factor de escalamiento y un contador. Primeramente, el factor de escalamiento es la relación entre la frecuencia de entrada y la frecuencia de salida deseada:

$$Escala = \frac{f_{\text{entrada}}}{f_{\text{deseada}}}$$

Asumiendo que tenemos una frecuencia de 50MHz y deseamos una salida de 200Hz, tenemos que:

$$Escala (\text{periodo}) = \frac{50\text{MHz}}{200\text{Hz}} = 250000$$

Por lo tanto, el contador para el divisor de frecuencia tiene como función generar la señal de salida de 200Hz cada 250000 ciclos.

El procedimiento realizado al momento de programar en VHDL el modulador con las funcionalidades mencionadas fue el analizado en el Laboratorio 1 de VHDL del Práctico de Electrónica Digital, en el transcurso de las páginas 23 a 27, bajo el encabezado de “Sistema 02”. A continuación se adjunta la imagen de la máquina de estados y tabla de verdad de la misma hoja de datos junto al diagrama de bloques realizados por quienes escriben.

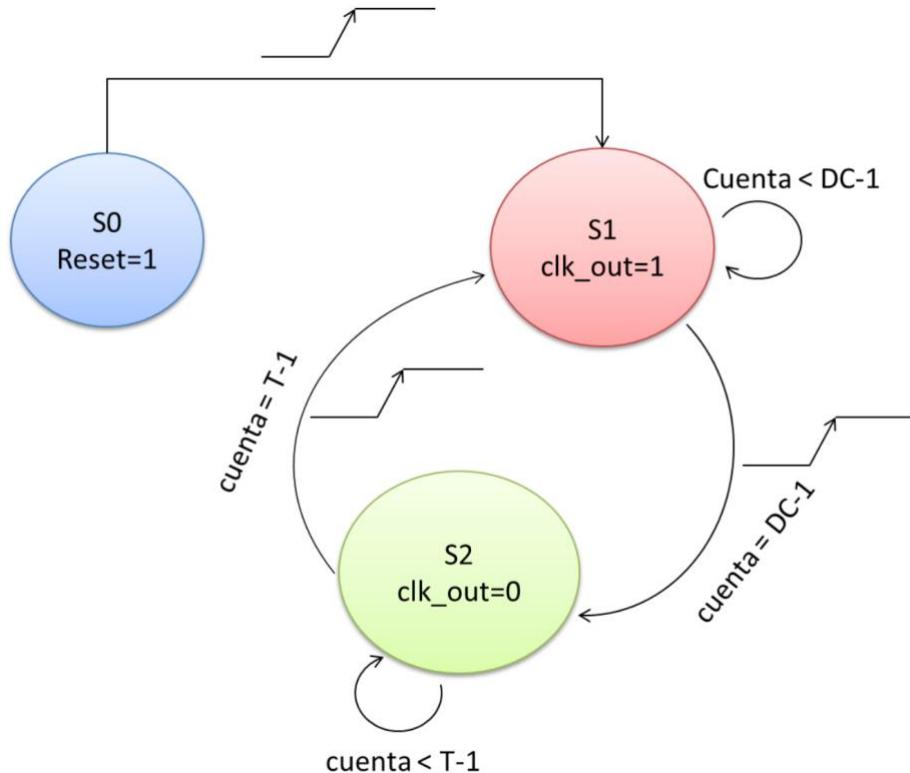


Figura 30. Diagrama de la máquina de estado usado para el modular a modo de divisor de frecuencia utilizado

S_i	$Q_1(n)$	$Q_0(n)$	$cuenta$	$D_1(n)$	$D_0(n)$	S_{i+1}
S_0	0	0	X	0	1	S_1
S_1	0	1	$< (DC - 1)$	0	1	S_1
S_1	0	1	$= (DC - 1)$	1	0	S_2
S_2	1	0	$< (T - 1)$	1	0	S_2
S_2	1	0	$= (T - 1)$	0	1	S_1
S_3	1	1	X	0	0	S_0

Tabla 12. Tabla de Verdad o Lógica referente a la máquina de estado que posibilita el funcionamiento de los moduladores

Desarrollando por suma de productos:

$$D_1 = S_1(\text{cuenta} = DC - 1) + S_2(\text{cuenta} < T - 1)$$

$$D_0 = S_0 + S_1(\text{cuenta} < DC - 1) + S_2(\text{cuenta} = T - 1)$$

$$\text{clk}_{out} = S_1$$

$$\text{Reset} = S_0 + S_1(\text{cuenta} = T)$$

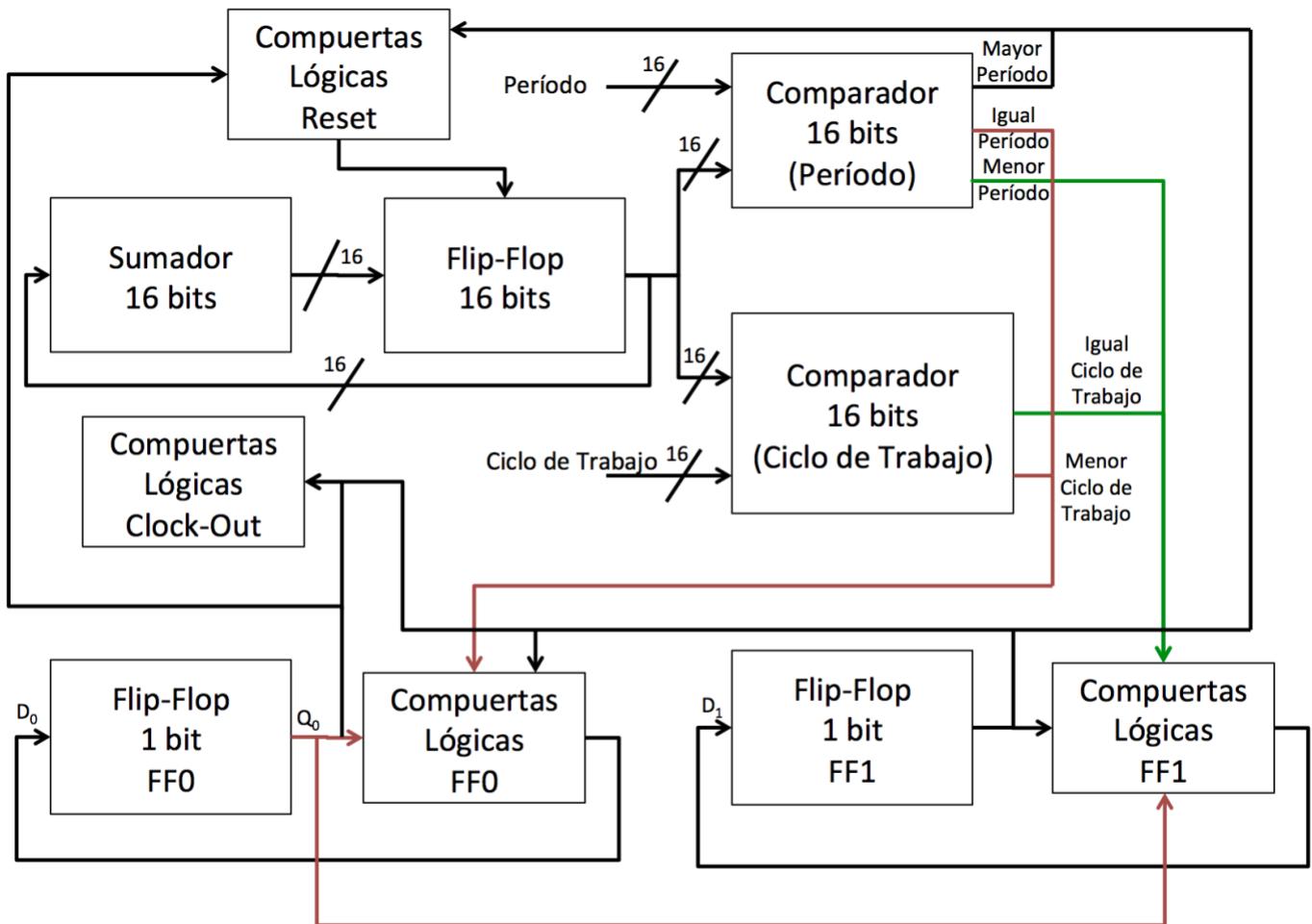


Figura 31. Diagrama de bloques de un modulador arbitrario, consecuencia de la máquina de estado anterior

8.2. Modulador de 10 Hz

El modulador de 10 Hz fue necesario realizarlo debido a que en numerosas partes del trabajo se precisó la implementación de una cuenta en decimal, base para posteriores cuentas en segundos y decenas de segundos logradas con la concatenación de contadores ya explicitadas. Por ello, al precisar una frecuencia cuyo respectivo período sea de 100 ms (equivalentemente a 0.1 segundos), y al ser la frecuencia el inverso del período, se concluye la creación de un divisor de frecuencias de 10 Hz.

De esta manera y análogamente al ejemplo planteado anteriormente, se prosiguió a hallar el factor de escalamiento para reducir el período de la señal y en base al mismo hacer lo propio con el ciclo del trabajo de la misma. Para ello, al haber realizado un divisor de frecuencia de 16 bits, el máximo factor con el que se pudo haber disminuido el período era de 65535 ($2^{17}-1$). Habiendo considerado esta última aclaración, se prosiguió a hallar el mencionado factor:

$$Escala_1 = \frac{f_{entrada}}{f_{final}} = \frac{50 \text{ MHz}}{10 \text{ kHz}} = 5000 \Rightarrow (5000)_{10} = (1001110001000)_2$$

$$Escala_2 = \frac{f_{entrada}}{f_{final}} = \frac{10 \text{ kHz}}{10 \text{ Hz}} = 1000 \Rightarrow (1000)_{10} = (1111101000)_2$$

De la misma manera, se decidió programar al modulador de tal manera que el ciclo de trabajo resultante luego de cada divisor de frecuencia sea la mitad del período, de tal modo de conseguir un tiempo de estado alto igual al tiempo de estado bajo (considerando el error brindado por la placa y demás posibles ocurrencias prácticas que imposibiliten una completa exactitud en las medidas).

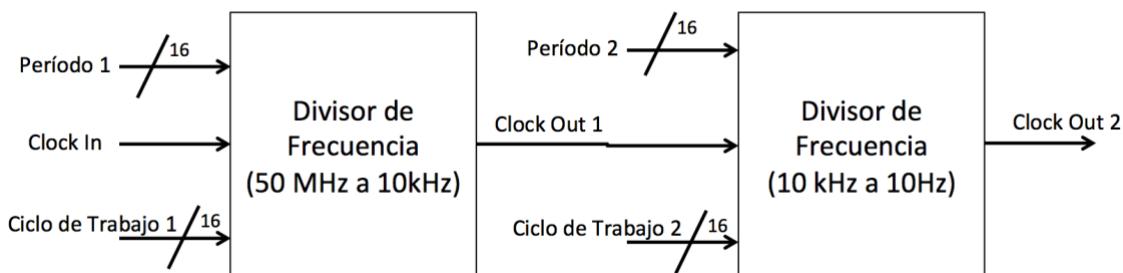


Figura 32. Diagrama de bloques de un modulador de 10 Hz

8.3. Modulador de 0.5 Hz

Al decidir programar un nuevo modulador de señal que determine los pares de segundos (es decir, 2 segundos de período) para su uso exclusivo en la Pantalla 4, donde se solicitaba la variación de la información desplegada en los *displays* cada 2 segundos, se prosiguió de una manera completamente análoga a la descrita para el caso del modulador de 10 Hz.

$$Escala_1 = \frac{f_{\text{entrada}}}{f_{\text{final}}} = \frac{50 \text{ MHz}}{10 \text{ kHz}} = 5000 \Rightarrow (5000)_{10} = (1001110001000)_2$$

$$Escala_2 = \frac{f_{\text{entrada}}}{f_{\text{final}}} = \frac{10 \text{ kHz}}{0.5 \text{ Hz}} = 20000 \Rightarrow (2000)_{10} = (100111000100000)_2$$

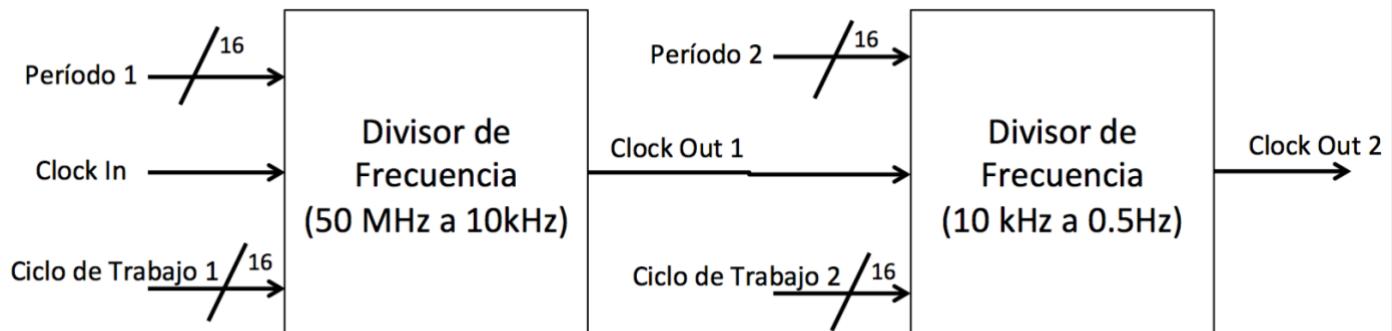


Figura 33. Diagrama de bloques de un modulador de 0.5 Hz

8.4.Modulador de 2 Hz

Finalmente, se buscó programar un nuevo modulador de señal que avise cada medio segundo (o sea, 0.5 segundos de período) para su uso exclusivo al desplegar el mensaje de error (letra “E”), donde se solicitaba la variación de la información desplegada en los *displays* cada 0.5 segundos. De esta forma, se prosiguió de una manera completamente análoga a la descrita para el caso del modulador de 10 Hz y 0.5 Hz.

$$Escala_1 = \frac{f_{entrada}}{f_{final}} = \frac{50 \text{ MHz}}{10 \text{ kHz}} = 5000 \Rightarrow (5000)_{10} = (1001110001000)_2$$

$$Escala_2 = \frac{f_{entrada}}{f_{final}} = \frac{10 \text{ kHz}}{2 \text{ Hz}} = 5000 \Rightarrow (5000)_{10} = (1001110001000)_2$$

Nótese que en ambos casos se pudo haber realizado con más instancias de moduladores, pero no mediante la utilización de un solo modulador, puesto que el factor de escalamiento en las situaciones estudiadas resultarían mayores que el máximo mencionado soportado por 8 bits de información. Debido a ello, se optó por la versión que usó menos recursos de la placa, dejándolos disponibles para otros posibles futuras utilidades.

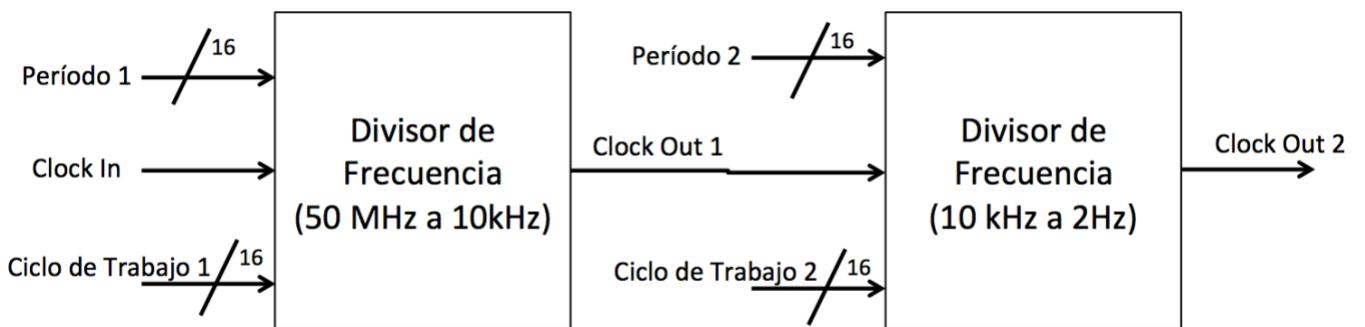


Figura 34. Diagrama de bloques de un modulador de 2 Hz

9. Macro Bloques

9.1. Macro Programa

Este bloque, considerado como macro bloque debido al hecho que engloba numerosos bloques previamente realizados y analizados, considerados así bloques “más pequeños”, consiste en la distribución de la información recibida por la mezcladora respecto a cada programa y por tanto, en estrecha relación con el macro bloque de Macro Pantalla (descrito a continuación) respecto a los datos mostrados en los *displays*. De esta manera, este bloque contiene una instancia de cada uno de los ocho programas realizados, con sus respectivas entradas (correspondientes al tiempo actual y al clock de la FPGA de 50 MHz) y salidas (las acciones realizadas por el programa en el tiempo recibido como entrada).

Así, teniendo cada programa las cinco salidas adecuadas a las cinco posibles acciones para cada mezcladora (llenar, vaciar, calentar, comprimir y girar), se prosigue a conectar cada una de las mismas a un multiplexor (necesitando por tanto cinco multiplexores) de 16 entradas a 1 salida con cantidad de bits variantes dependiendo de la cantidad de bits de la información de salida del programa, cuyos cuatro bits de control se correspondan con el vector de entrada del Macro Programa que define el número del programa a realizar.

Nótese que al tener que utilizar cuatro bits de control con su consiguiente cantidad de entradas, y al poseer solo 8 programas que ligar, se debieron conectar la mitad de las entradas del multiplexor a Tierra (0 Voltios), puesto que estas nunca se usarían al solo usarse las entradas 1 a 8 del dispositivo. De esta manera y acorde a lo que se mencionó, los multiplexores de comprimir, vaciado y calentar ostentan un bit de información, mientras que el multiplexor de girar posee 2 bits y el multiplexor de llenado tiene 4 bits. Una vez multiplexada la información al recibir la información de que programa es el que se debe desplegar, siendo señaladas por las salidas de los multiplexores, son aquellas señales que salen del macro bloque para su posterior recibimiento por el macro bloque de Macro Pantalla, el cual rige la información a mostrar en los *displays* de siete segmentos. Finalmente, es relevante

remarcar que la cuenta del tiempo por la que se rige el Macro Programa se encuentra instanciada de forma externa al mismo, siendo esta la misma que aquella que entra a la Macro Pantalla.

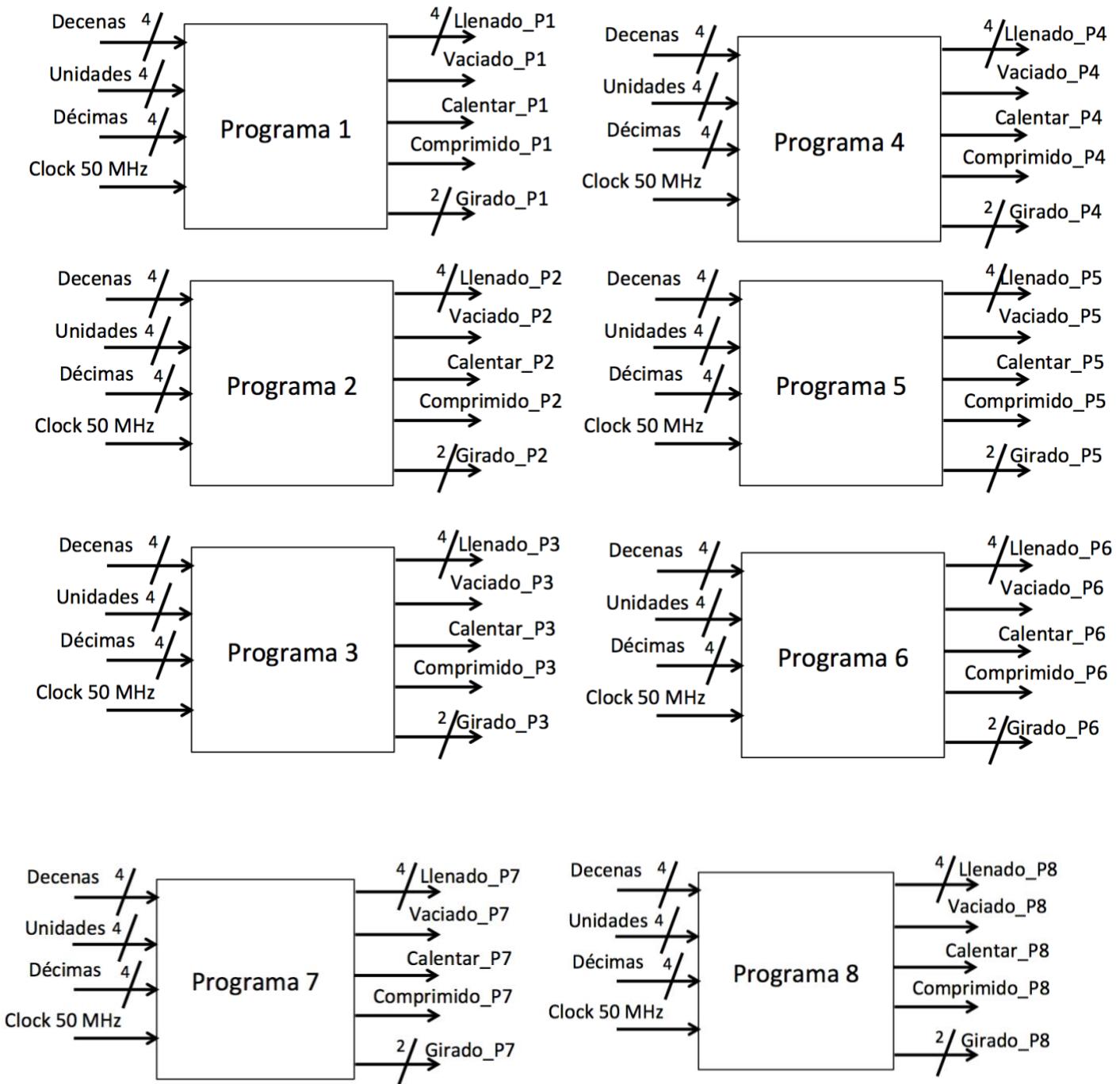


Figura 35. Diagrama de bloques indicando las instancias de cada Programa con sus respectivas entradas y salidas

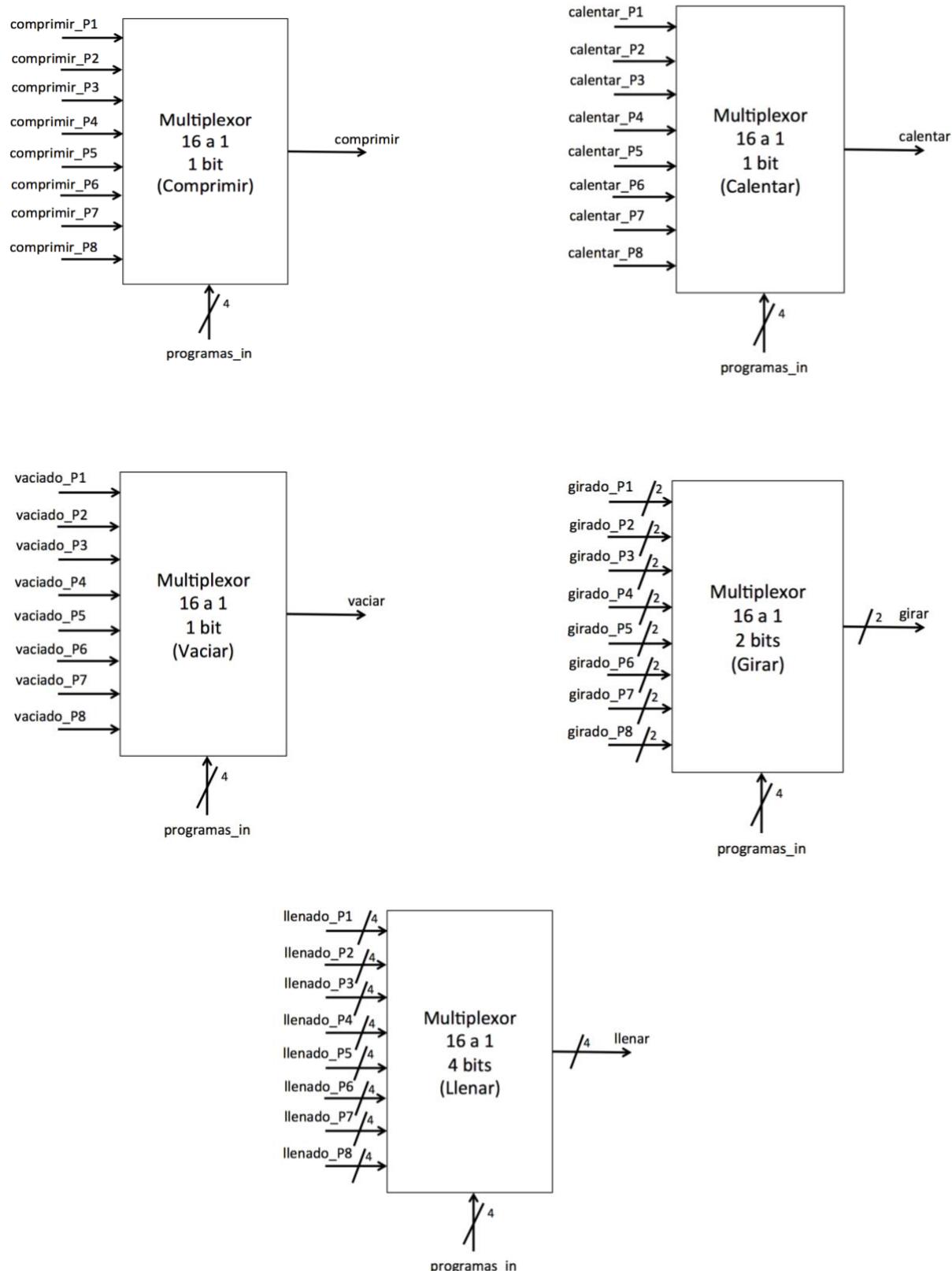


Figura 36. Diagrama de bloques indicando la multiplicación de las señales necesarias para el Macro Programa recabadas por los Programas

9.2.Macro Pantalla

Continuando el estudio de los macro bloques, se programó aquel análogo al mencionado del Macro Programa pero que se encargara de almacenar y administrar la información desplegada en los cuatro *displays* de 7 segmentos según los formatos previamente analizados de Pantalla 1, Pantalla 2, Pantalla 3 y Pantalla 4.

De esta forma, se prosiguió a instanciar cada una de las reiteradas pantallas dentro del macro bloque y, mediante el uso de correspondientes multiplexores, decidir que información mostrar en cada momento, a modo de condicional en el caso que ocurriera cierta acción del sistema. Estos multiplexores fueron cuatro diferentes instancias del mismo tipo de multiplexor de 4 entradas a 1 salida de 8 bits de control, puesto que cada cual determinan que mostrar en cada uno de los cuatro *displays* de la FPGA. Los correspondientes dos bits de control resultaron ser en todos los dispositivos el del vector de switchs, los cuales correspondían a la activación o desactivación de los switchs 8 y 9 encargados de regular y cambiar la pantalla actualmente enseñada y desplegada en los *displays*. La cuenta de tiempo entrante a la Pantalla 1 se encuentra totalmente sincronizada a la existente en los programas debido a que son ambas las mismas señales, entrantes a través del macro bloque de Mezcladora a partir del mismo contador decimal y modulador de 10 Hz respectivo. En el caso de haberse necesitado otra pantalla, tan solo se hubiese programado de forma externa como interna al macro bloque y posteriormente instanciado en el mismo, debiendo agregar una nueva entrada a cada multiplexor de tal manera de posibilitar la muestra en pantalla del nuevo formato necesario.

Así, teniendo como entrada del macro bloque estudiado a cada una de las entradas necesarias para cada pantalla y como salida los cuatro displays de 7 segmentos a conectar y asignar directamente a los pines brindados por el *datasheet* de la FPGA (de modo similar a lo ocurrido en el Macro Programa), se continuó a instanciar este programa VHDL en el programa Mezcladora analizado a continuación. Nótese que este proceso de instanciar se pudo haber evitado al desplegar todos los dispositivos de las Pantallas (y su correspondiente programación) directamente en el bloque tratado, de la misma forma que se pudo haber sorteado el instanciar cada programa en el Macro Programa al desplegar la información de cada programa en tal macro bloque, procesos que se tuvieron en consideración pero fueron inmediatamente descartados debido a la obvia y absoluta desprolijidad de

código consecuencia del mismo, lo que podría también concluir en un mal funcionamiento del programa general creado.

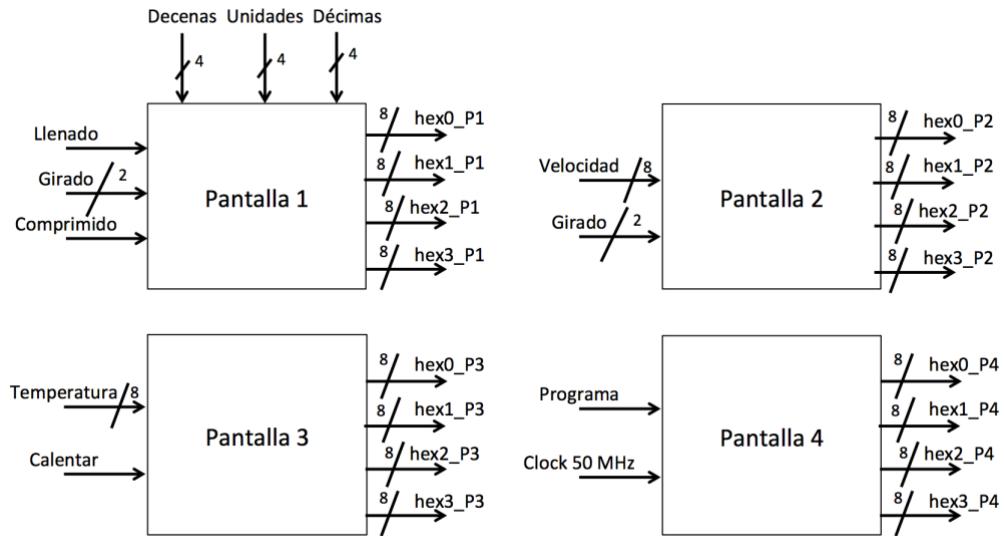


Figura 37. Diagrama de bloques indicando las instancias de cada Pantalla con sus debidas entradas y salidas

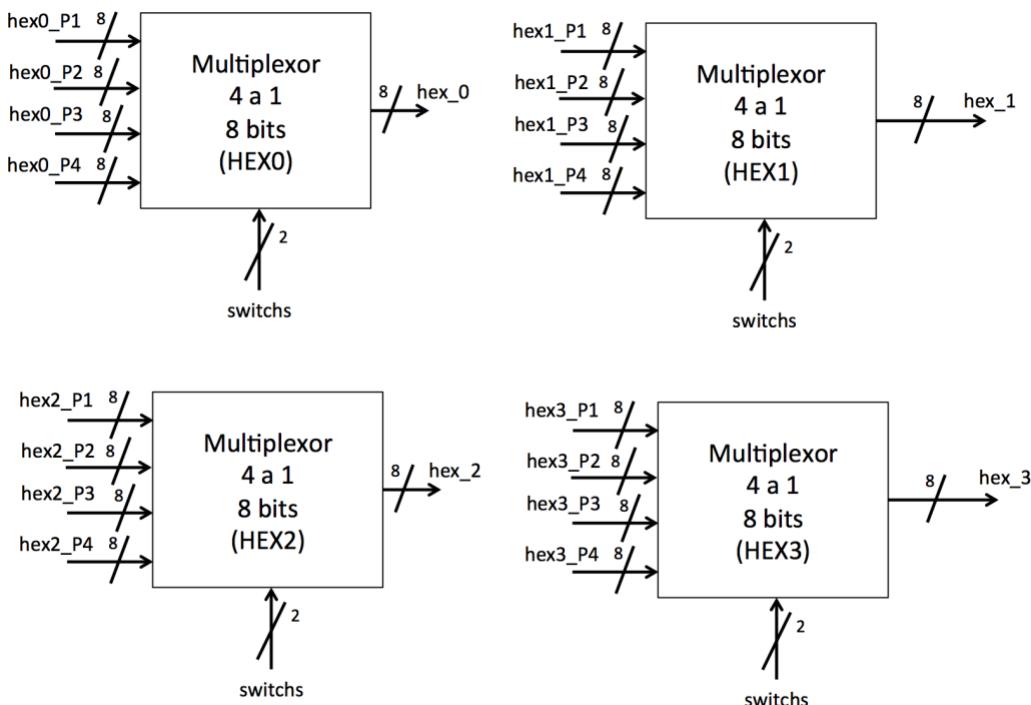


Figura 38. Diagrama de bloques mostrando la multiplexación de las señales recabadas por las Pantallas

9.3. Mezcladora

El macro bloque tratado a continuación es el encargado de concatenar el Macro Programa con la Macro Pantalla, creando de manera implícita las características de cada una de las cuatro mezcladoras a instanciar en el subsiguiente macro bloque de mayor jerarquía, nombrado Obligatorio 2 (siendo este el *Top-Level Entity* del trabajo). Es así que el bloque estudiado posee una instancia de Macro Programa y una instancia de Macro Pantalla junto a una instancia de Direccionar (encargado de decodificar la información recibida respecto del primer byte recibido o byte de direccionamiento) y Comandar (responsable de decodificar el segundo y tercer byte correspondiente a los bytes de comando y datos respectivamente), siendo los demás dispositivos usados de manera auxiliaría para definir para qué estado del sistema en el momento del actuar se debe activar o mostrar o transmitir la información necesitada.

Es de esta manera que es aquí donde se decidió instanciar el contador principal que regula el funcionamiento de los programas en conjunto con las pantallas, regulando el transcurso del tiempo en décimas, unidades y decenas de segundo al recibir un reloj modulado a 10 Hz, todo programado de la manera ya explicada. Las salidas de este contador decimal, correspondientes a las tres respectivas unidades mencionadas son conectadas a las entradas adecuadas del Macro Programa al igual que a las de la Macro Pantalla, compartiendo de esta manera la contabilidad del tiempo que ha acontecido. Este contador es también el encargado de pausar el programa configurado y en ejecución al activar el *switch* 7 definido por la letra del Obligatorio 2. Esto fue posible al conectar el *Carry In* a la señal recibida de este *switch*, debiendo funcionar al tener el *switch* señal lógica 0 (y por tanto el *Carry In* poseer señal lógica 1) y detener la cuenta en caso contrario, pudiendo continuarla al desactivar nuevamente el *switch* mencionado. Nótese que el contador decimal posee salidas tanto en codificación binaria como decimal para facilitar la muestra en la Pantalla 1 del tiempo acontecido y ahorrar la utilización innecesaria de excesivos decodificadores de displays de 7 segmentos. Sin embargo, esta ingeniosa estrategia no fue utilizada debido a la falta de tiempo para adecuar el código a un formato que utilice aún menos recursos de los ya ahorrados con otras ideas planteadas.

A su vez, como se puede observar en los diagramas de bloques o en el código respectivo, a fin de que los programas (regulados por el Macro Programa) no inicia antes de la señal de *start* o no continúe luego de la señal de *stop*, se implementó una condición fácilmente detectable debido a su extensión en

el código del programa de modo que el reloj del macro bloque funcione en el rango debido. Esta condición, al incluir numerosos elementos de diferentes bloques tanto externos como internos a Mezcladora, posee grandes posibilidades de mal funcionamiento, si bien como se muestra en el video demostrativo respectivo no se encontró problema al respecto.

Finalmente, se hizo uso de un multiplexor de 2 bits con 16 entradas a 1 salida, cuyos 4 bits de control se correspondan con los 4 switchs encargados de regular la mezcladora con la cual se trabaja en el momento (referidos a los switchs 3, 4, 5 y 6). Como entradas, el mismo posee el número distintivo de cada mezcladora (siendo el “00” referente a la mezcladora 1, “01” a la mezcladora 2, “10” a la mezcladora 3 y “11” a la mezcladora 4), la cual es elegida condicionalmente por la activación de los switchs y cuya señal es conectada a un comparador de 2 bits quien la compara a la señal entrante desde el macro bloque del Obligatorio 2, característica de cada instancia de mezcladora presente en el mismo. La salida del comparador al buscar por igualdad es la acarreada hasta una de los componentes que intervienen en las condiciones del reloj del Macro Programa, correspondiente a la acción solicitada de poder elegir un programa a desplegar de manera manual.

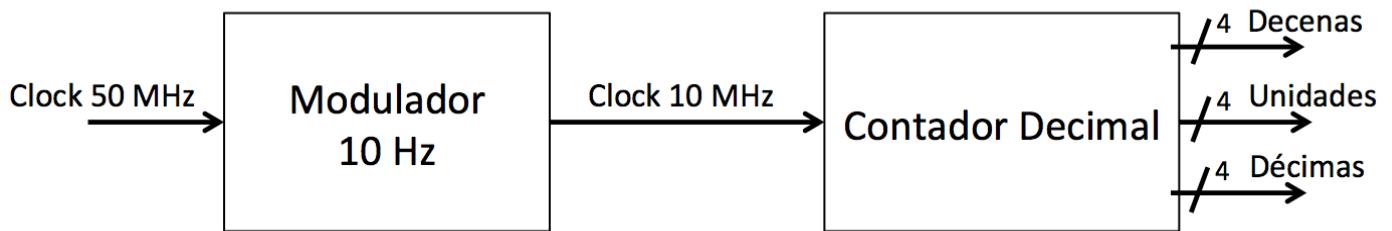


Figura 39. Diagrama de bloques mostrando los componentes que posibilitan el contar la cantidad de tiempo transcurrido

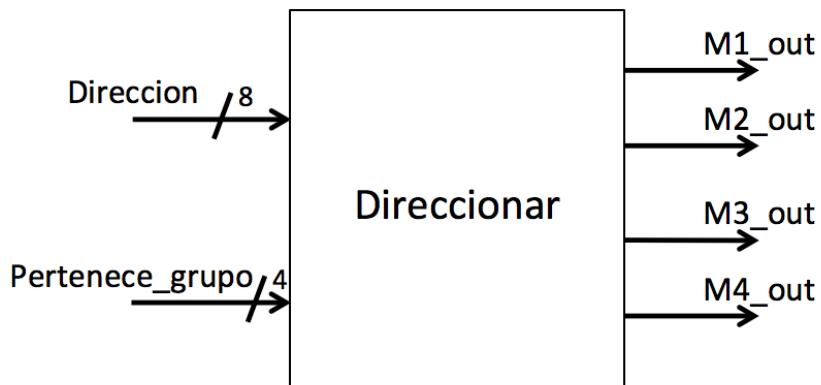


Figura 40. Diagrama de bloques del boque de Direccinar con sus entradas y salidas a modo de lo explicado en el punto 4.

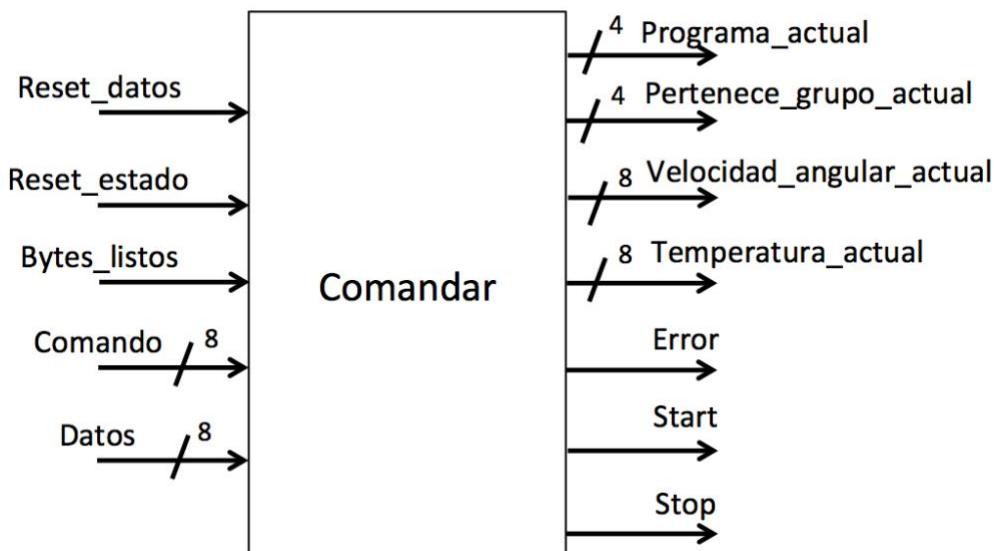


Figura 41. Diagrama de bloques del boque de Comandar con sus entradas y salidas a modo de lo explicado en el punto 4.

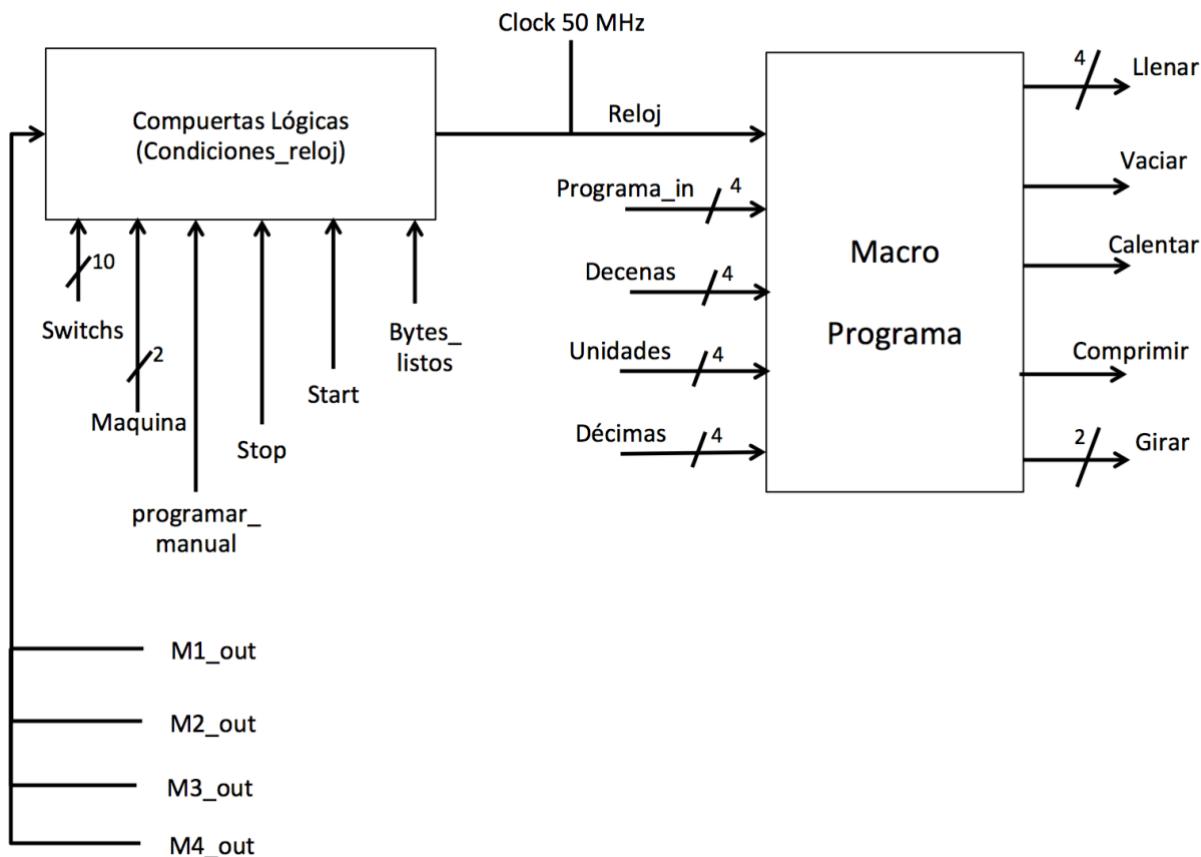


Figura 42. Diagrama de bloques mostrando la interconexión entre señales anteriores y la instancia de Macro Programa creado en la Mezcladora

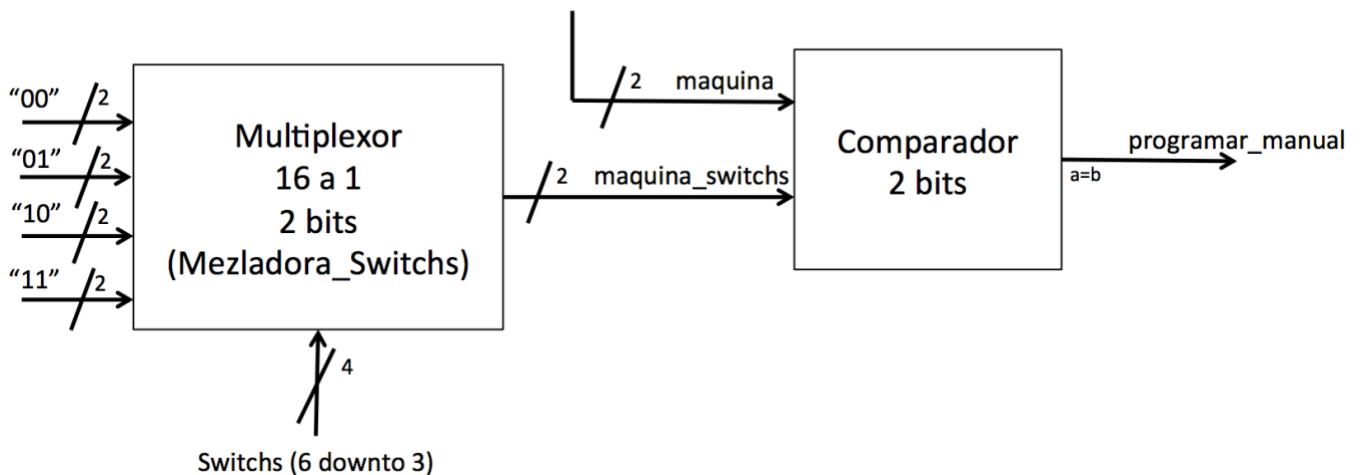


Figura 43. Diagrama de bloques de la sección de Mezcladora encargado de multiplexar las señales de las máquinas de acuerdo a lo seleccionado por los switchs mediante acción del usuario

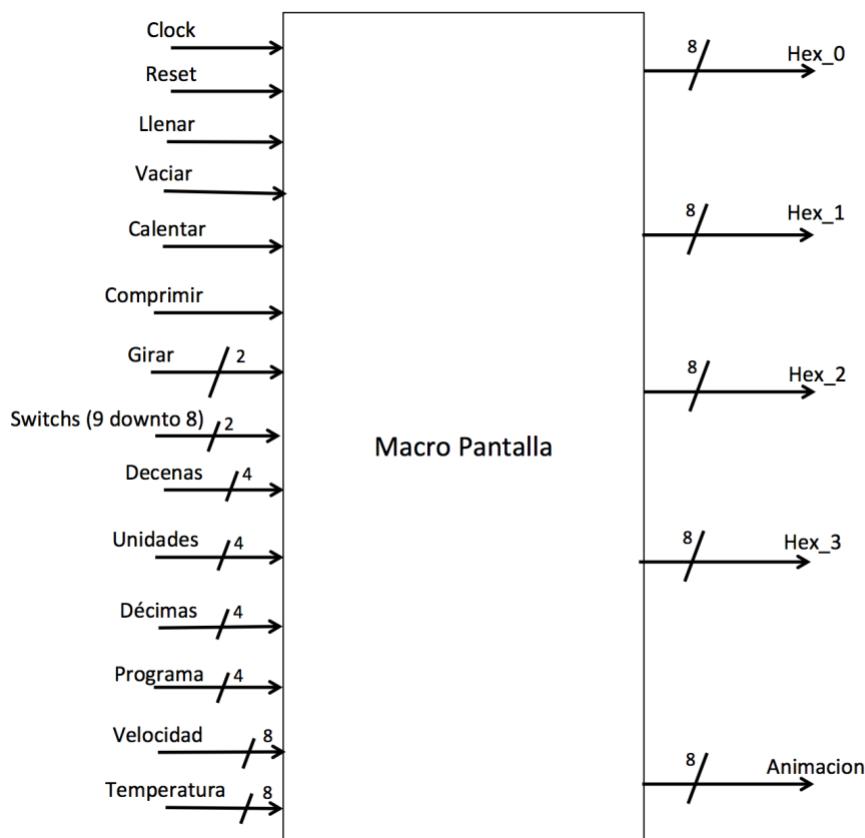


Figura 44. Diagrama de bloques de la instancia creada dentro de Mezcladora del bloque de Macro Pantalla, con la explicitación de sus merecidas entradas y salidas

9.4. Obligatorio 2 (*Top-Level Entity*)

Concluyendo el análisis de los macro bloques, se estudia la creación del bloque de mayor entidad y jerarquía, englobando todos los programas realizados y descritos en este documento, y considerado y definido consecuentemente como la *Top-Level Entity* del Obligatorio, a partir del cual se asignan todos los pines a utilizar y se inicializa la totalidad del programa VHDL en la FPGA. En resumen, este bloque se encarga de organizar toda la información recibida por el bloque encargado de la decodificación de la transmisión serial RS 232 para la correspondiente y adecuada inicialización de las cuatro mezcladoras solicitadas y el posterior tratamiento de los valores y señales recibidas respecto a lo solicitado por el usuario que se encuentra controlando la placa y modificando el funcionamiento del programa a través del software de transmisión serial brindado por la facultad, y mediante los switchs y los botones de la FPGA para el despliegue y procesamiento de información requerida.

De esta manera, y visto que existen muchos aspectos por analizar, se prosigue a la explicación gradual del procedimiento realizado desde lo más específico y particular hasta lo más general e interconectado. Comenzando así por el despliegue del mensaje de error en caso de una parada de emergencia (señal de stop) o si existe un error con la temperatura (señal de error), se prosiguió a crear una pequeña maquina de estados de 2 estados regulados por el reloj modulado de 2 Hz programado y explicado anteriormente, señal que sirviera de bit de control de un multiplexor de 8 bits de 2 entradas a 1 salida y cuyas entradas se correspondiesen con la acción de no mostrar nada y mostrar la letra “E” solicitada de error, a efectos que la esta letra se mostrara (“destellara”) en los *displays* dos veces por segundo. Este multiplexor, poseyendo ya datos codificados para la muestra en los *displays*, conecta su salida con el formato del mensaje de error a otros cuatro multiplexores (uno para cada mezcladora) de las mismas características, cuyo bit de control sea activo al obtener cualquiera de las dos señales de error posibles desde las mezcladoras. La entrada faltante de estos multiplexores son así la señal de la animación actual de cada mezcladora, lográndose de esta forma lo solicitado del despliegue general de información de las cuatro mezcladoras en caso que el *switch 0* posea una señal lógica 0 (esté desactivado), por lo que se debió desarrollar la salida de los cuatro multiplexores a los cuatro *displays* de 7 segmentos existentes en la FPGA. Estas señales se indican en el diagrama de bloques siguiente como “hex0_general”, “hex1_general”, “hex2_general” y “hex3_general”.

A continuación, se prosiguió a realizar cuatro instancias del bloque Mezcladora que se correspondieran con las cuatro mezcladoras pedidas para el trabajo. Luego de crear una señal del VHDL para cada salida de cada mezcladora, se prosiguió a conectar las respectivas señales a ocho diferentes multiplexores de 16 entradas a 1 salida con cantidad de bits variables cuyos bits de control sean los 3 *switchs* indicados para determinar el programa deseado a comenzar su ejecución (*switch 0*, *switch 1* y *switch 2*) junto a un 0 referentes a valores no utilizados. Nótese de esta forma que no se utilizan todas las entradas del multiplexor, enviando todas las no utilizadas a Tierra, es decir, conectándolas a una señal lógica de 0 constante. De esta forma, se utilizaron multiplexores de 1 bit para los multiplexores de las señales de error, stop, comprimir, vaciar y calentar de las mezcladoras, a la vez que multiplexores de 2 bits para aquellas de girar y de 4 bits para las de llenar y programas.

Finalmente, para decidir que información de que mezcladora indicar en *display* se prosiguió a utilizar nuevamente multiplexores de 16 entradas a 1 salida de 8 bits de control (los cuales se conectan directamente a cada *display* de 7 segmentos, salidas del bloque del Obligatorio 2), necesitando uno para cada *display* existente en la placa FPGA. Los 4 bits de control del multiplexor fueron lógicamente de este modo el botón encargado del *reset* general del trabajo y los 3 *switchs* anteriores encargados de determinar el programa requerido a ejecutar (*switch 0*, *switch 1* y *switch 2*). Se puede observar que se encuentra considerándose el caso en que el *switch 0* posee valor lógico 0, con el cual como se mencionó debe mostrarse todas las mezcladoras en su respectivo proceso en el momento del accionar. Este caso, según la organización decidida para el trabajo, se debió pensar y agregar manualmente en las entradas de cada multiplexor, comprobando la acción realizada por la placa con todas las combinaciones posibles de las 4 señales presentes en los bits de control mencionados. Los datos del programa actualmente en funcionamiento junto con la mezcladora a la cual se le está aplicando se debió mostrar por defecto de una manera especificada por la letra, por lo que se precisó un decodificador del *display* de 7 segmentos que permitiera el decodificar la señal de salida del multiplexor de los programas para su correcto despliegue en pantalla.

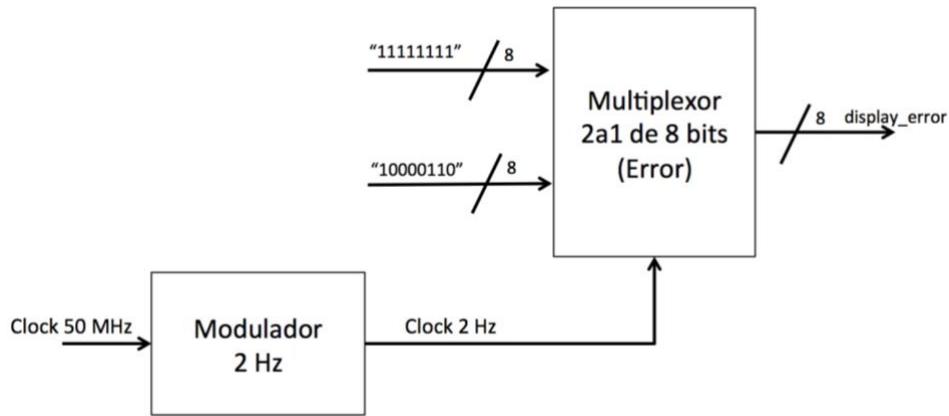


Figura 45. Diagrama de bloques que muestra las conexiones necesarias para posibilitar la máquina de estado con responsabilidad de desplegar el mensaje de error solicitado

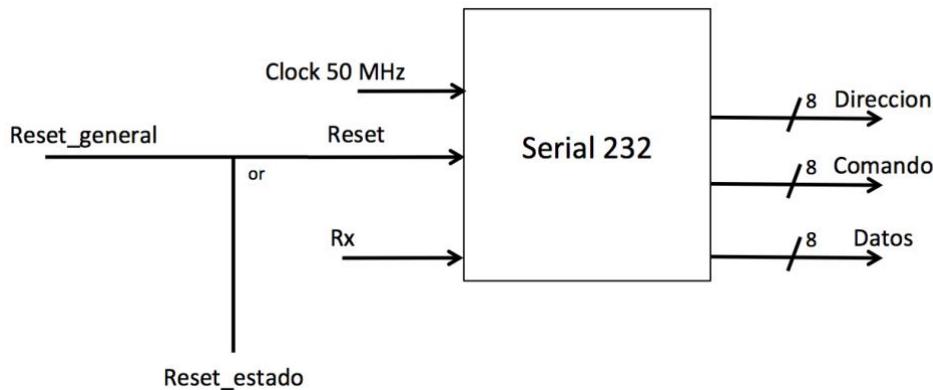


Figura 46. Diagrama de bloques del bloque del Serial 23 con sus adecuadas entradas y salidas

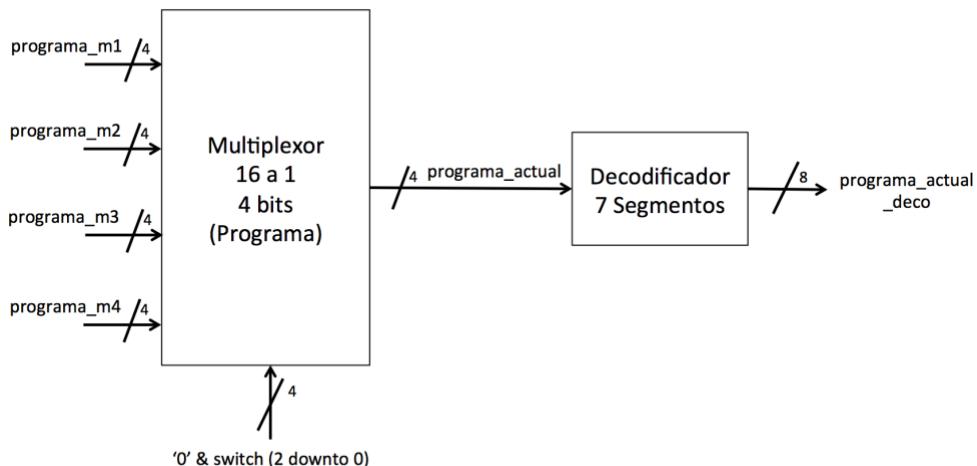


Figura 47. Diagrama de bloques indicando la multiplexación encargada de mostrar y definir el programa a ejecutar

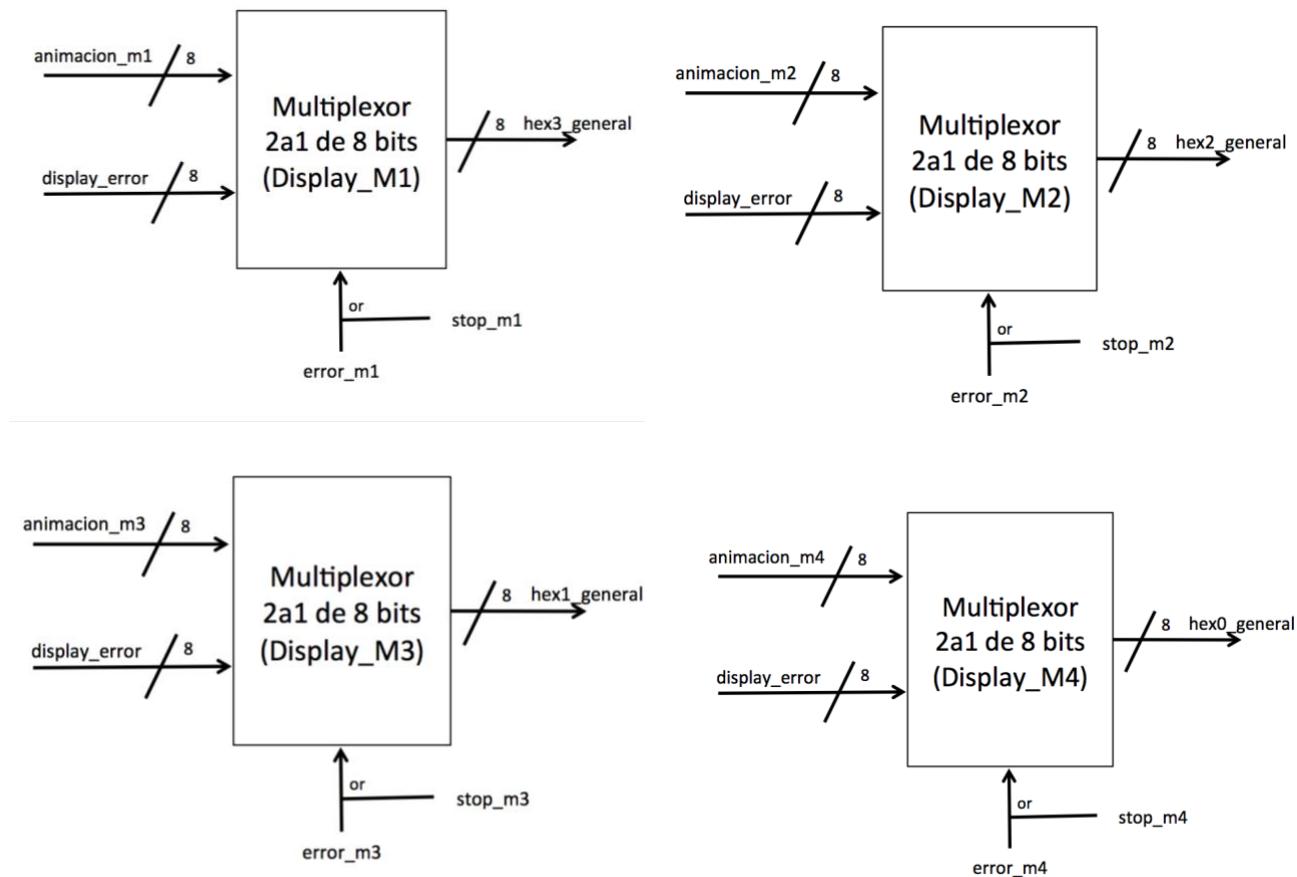


Figura 48. Diagrama de bloques exponiendo la multiplexación necesaria para decidir cuando indicar el mensaje de error o la animación correspondiente al indicar el despliegue de información general de todas las mezcladoras

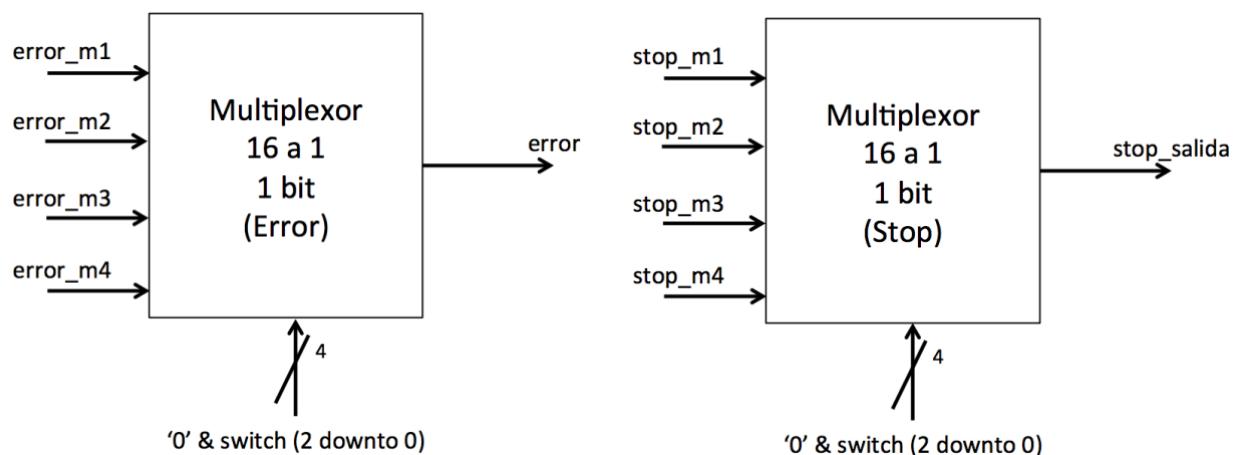


Figura 49. Diagrama de bloques de la multiplexación del mensaje de error de cada mezcladora en caso de temperatura elevada o parada de emergencia

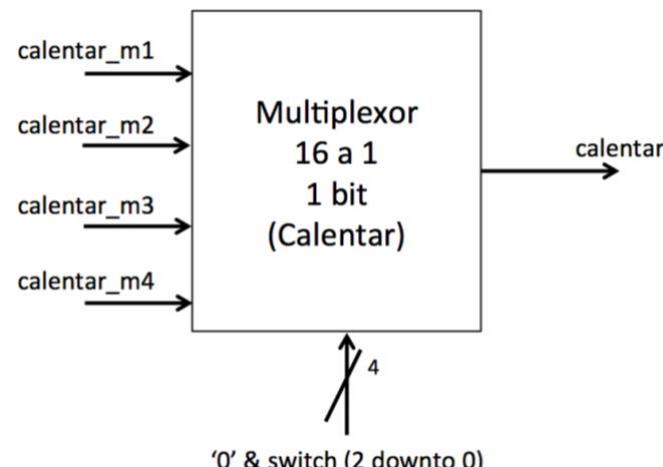
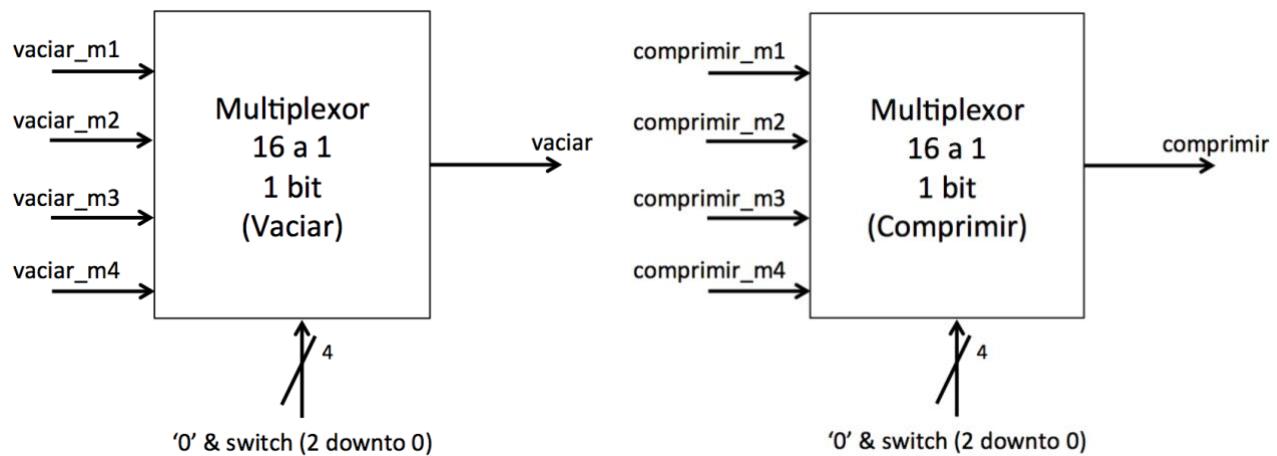
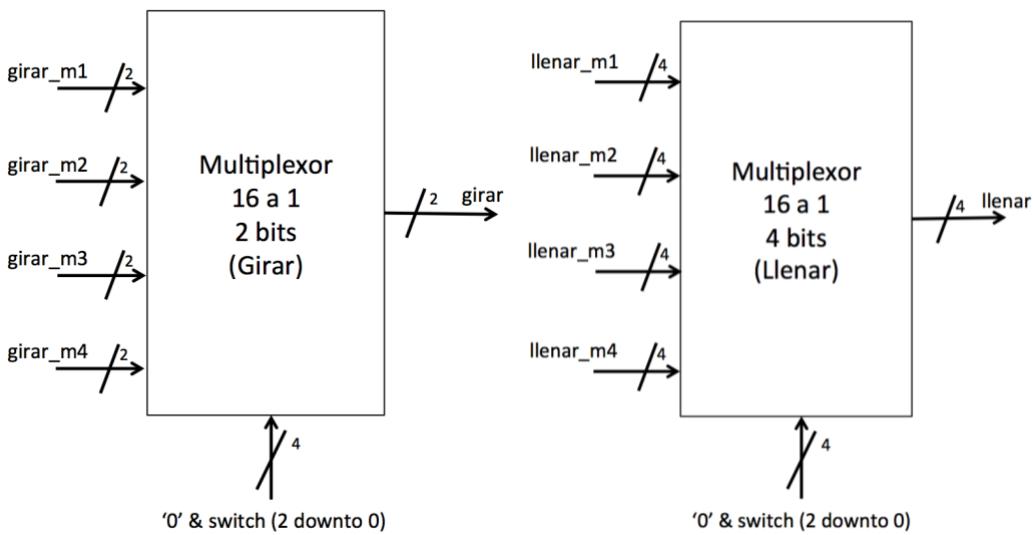


Figura 50. Diagrama de bloques de la multiplexación de señales respecto a aquellas acciones recibidas por cada mezcladora

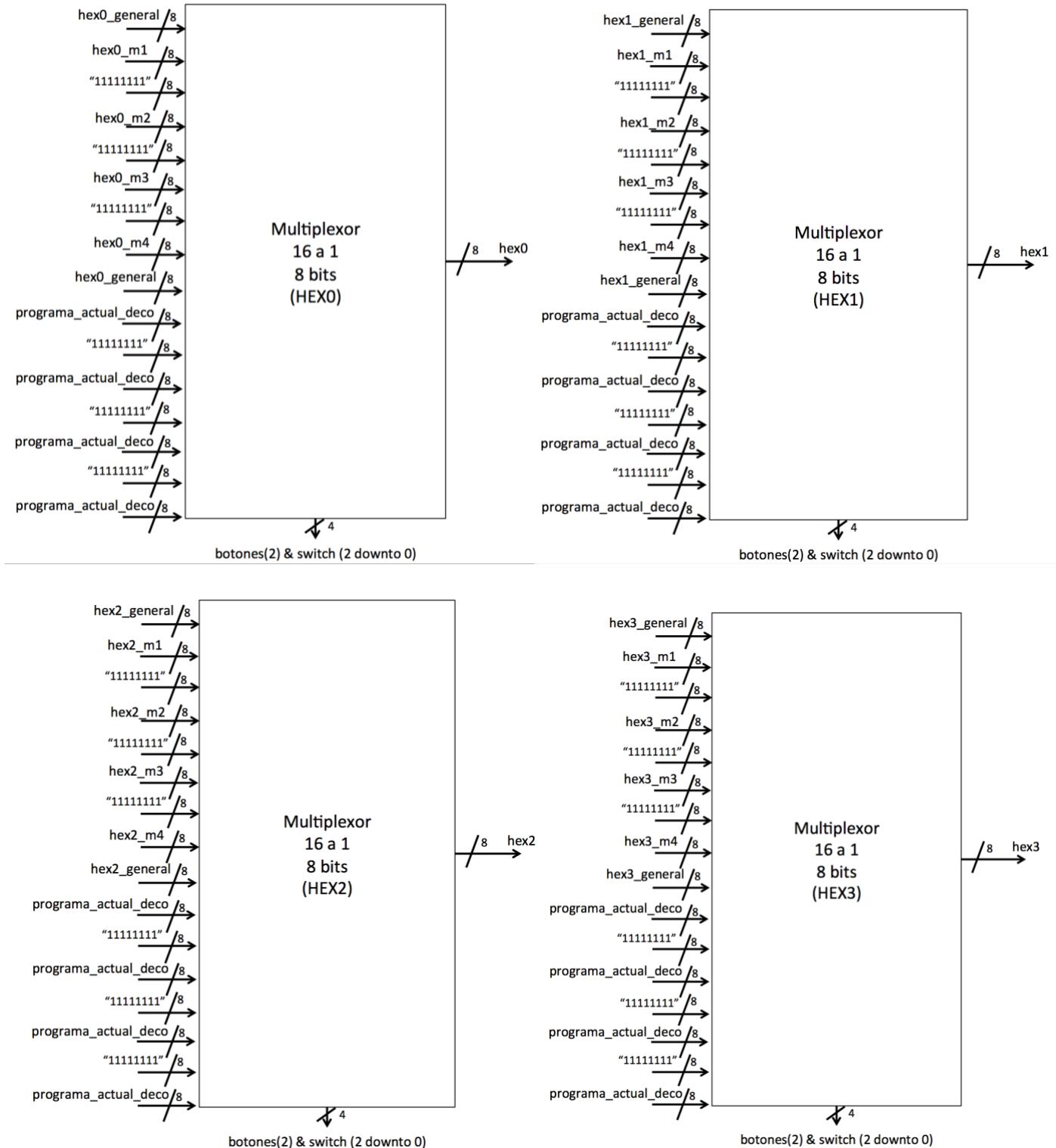


Figura 51. Diagrama de bloques indicando la relevante y global mutliplexación de señales, relevando la importante a cual mostrar según lo que el usuario elija mediante la activación o desactivación de los *switchs* conocidos

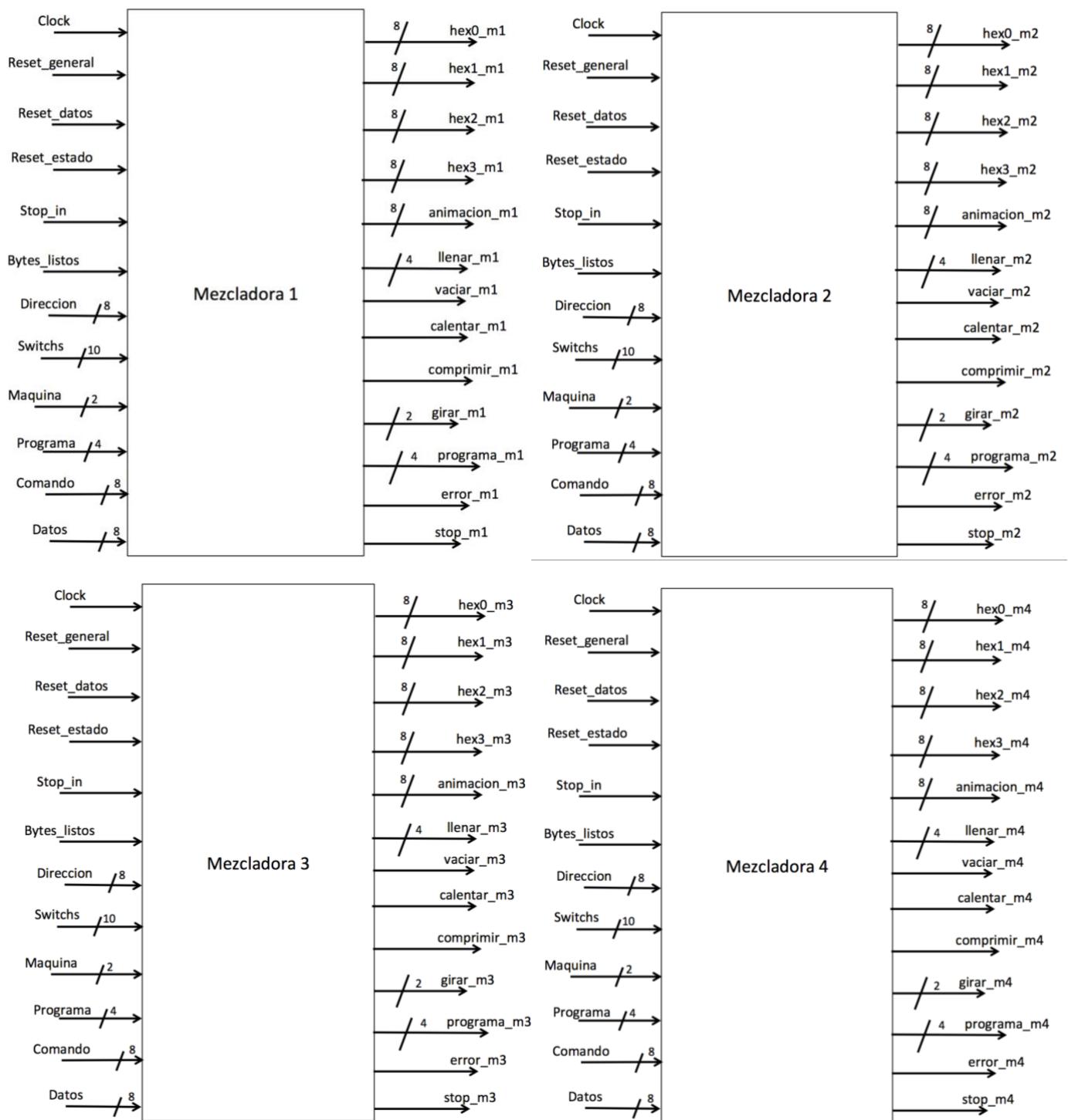


Figura 52. Diagrama de bloques de las cuatro instancias del bloque de Mezcladora con todas sus entradas y salidas

10. Especificaciones Finales

10.1. Multiplicador Binario

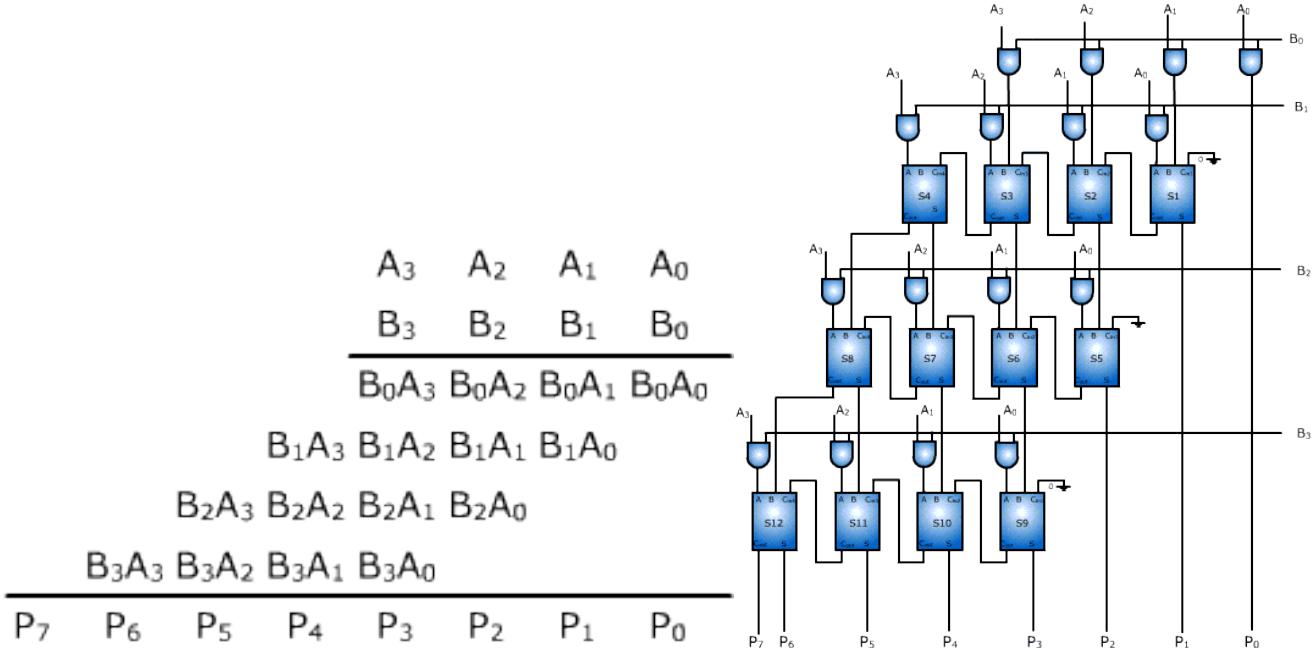


Figura 53. Diagrama de bloques del bloque de multiplicar de 4 bits necesario para determinar la velocidad de la mezcla

En el componente denominado “comandar”, encargado de la decodificación de los datos según el comando enviado y encargado además del almacenamiento de datos de cada mezcladora; dado que la velocidad angular recibida por la transmisión serial en 8 bits viene codificada de tal forma que la velocidad a utilizar en las mezcladoras es el producto de los 4 bits más significativos con los otros 4 bits menos significativos del byte recibido, se necesitó la implementación de un componente que realizara tal función.

Luego de investigar lo suficiente, se encontró la solución por hardware descrita en la anterior imagen que consta de varios sumadores de dos bits interconectados en cascada junto a varias compuertas AND. Este diseño proviene directamente de plantear la multiplicación de dos números binarios de 4 bits sin mayor complicación, sustituyendo la multiplicación de dos bits por compuertas and y la suma de dos bits por sumadores.

10.2. Animaciones en los *Displays* de 7 Segmentos

Se requiere indicar en los *displays* de 7 segmentos, ya sea el caso de Pantalla 1 o el *display* general de todas las mezcladoras, la animación de la acción actual en el programa. Pudiendo ser el caso del llenado (de cualquiera de los 4 líquidos), mezcla en sentido horario u antihorario, y finalmente compresión.

Guiados por la letra del obligatorio, dado que no se especifica nada más que la secuencia en que se deben encender los LEDS, se optó por un reloj de 10 Hz para controlar dicha secuencia ya que es una velocidad considerada adecuada para interpretar la secuencia a simple vista sin mayor problema.

Para la implementación en VHDL se usó una cadena de flip flops tal como en las registradoras (ver punto 4.2) en la que el reloj de los flip flops interconectados es el reloj de 2Hz, de las interconexiones de los flip flops se obtiene la señal de salida para los LEDS a encender y la salida del último flip flop es el encargado de resetear todos los flip flops, y por lo tanto apagar completamente el *display*. El ciclo se repite indefinidamente. En el caso que se necesitan encender dos LEDS a la vez como en el llenado y la compresión, se conectan ambas salidas a los LEDS en paralelo.

De esta manera, las animaciones en los *displays* de la FPGA funcionaron a base de mantener uno de los 8 LEDs de la mencionada pantalla siempre prendido (los 7 LEDs del *display* de 7 segmentos junto con el punto), siendo este variante respectivo a cual sea la animación a desplegar. La secuencia de LEDs a prender debió ser determinada e ingresada manualmente para cada caso en particular, variable del problema a solucionar. De este modo, en el caso de la animación llenado, se mantuvo siempre prendido el LED 3 de la placa, continuando por los LEDs 2 y 4, luego por el LED 6, siguiendo por los LEDs 1 y 5 y finalizando con el encendido del LED 0, proceso realizado de manera cíclica (ver punto 1.4 para la explicitación de la forma de enumerar los LEDs de un *display* de 7 segmentos utilizado). En el caso análogo y en cierta forma simétrico de la animación compresión, se debió mantener siempre prendido el LED 0 de la placa, continuando por los LEDs 1 y 5, luego por el LED 6, siguiendo por los LEDs 2 y 4 y finalizando con el encendido del LED 3, proceso también realizado de manera cíclica. Al realizar el caso quizás mas simple de la animación girado horario (debido al hecho de nunca tener que primeramente encender dos LEDs de forma simultánea), se tuvo que mantener siempre prendido el LED 0 de la placa, continuando por el LED 1, luego por el LED 2, siguiendo por el LED 3, para posteriormente encender el LED 4 y finalizando con el encendido del LED 5. Finalmente,

para el caso de animación girado antihorario, simétrico al del girado horario, se necesitó mantener siempre prendido el LED 0 de la placa, continuando por el LED 5, luego por el LED 4, siguiendo por el LED 3, para posteriormente encender el LED 2 y finalizando con el encendido del LED 1.

10.3. Botones Sin Retención

Se requiere el uso de los 3 botones sin retención de la placa para la realización del obligatorio. Las funciones de los botones son:

- El botón 2 es el encargado en el caso de estarse mostrando en ese momento la información de una mezcladora en particular, mientras se mantenga presionado, se debe mostrar el programa configurado en dicha mezcladora, de lo contrario, si lo que se muestra es el *display* general de las 4 mezcladoras, éste no tiene ninguna funcionalidad asignada.
- El botón 1 se encarga de resetear el sistema completo, es decir, eliminar la información asignada previamente a cada una de las mezcladoras, ya sea la configuración del programa, temperatura, velocidad y grupos.

En este caso, dado que no está indicado, para evitar complicaciones dadas por la eliminación de la configuración actual, también se decidió que el botón, además, finalice cualquier programa en progreso en cualquiera de las mezcladoras.

- El botón 0 es una parada de emergencia de todo el sistema, es decir, se finalizan los programas activos en cualquiera de las mezcladoras (quizá por un fallo técnico en una de las máquinas o un operario se encuentre en peligro).

Cabe destacar que a diferencia del botón 2, el botón 1 y el 0 tienen funcionalidad constante, es decir, cualquiera sea el estado de las mezcladoras o cualquiera sea la Pantalla que se esté mostrando en ese instante, los botones deben realizar la tarea asignada al presionarse.

11. Tabla de Prueba

Objetivos	Descripción	Testeo / Prueba	Observaciones
Obtención de datos mediante la Transmisión Serial del RS 232 de forma satisfactoria	Se debía ingresar en una aplicación programada en Java disponible en el sitio de Aulas de la facultad donde se digitaba información (números) correspondientes a 3 bytes y debieron ser recibidos correctamente por la Placa FPGA de Altera.	Ver video demostrativo en la carpeta de nombre "transmision serial", incluidos una recepción funcional y errores obtenidos durante el proceso a llegar a la misma.	En el video mencionado se hace muestra de que, una vez ingresados los 3 bytes al sistema a través del programa Java y el cable de transmisión serial, se despliegan adecuadamente los datos en los <i>displays</i> de 7 segmentos de la Placa.
Funcionamiento adecuado del primer byte recibido correspondiente al byte de Direcciónamiento solicitado.	El primer byte era el encargado de establecer a que mezcladora se le aplican los comandos y datos a especificar a continuación.	Ver video demostrativo en la carpeta de nombre "direccionar".	Las mezcladoras direccionadas mediante los bits seleccionados con los <i>switchs</i> se encuentran desplegados en los LEDs.
Creación y utilización satisfactoria del multiplicador binario.	A fin de multiplicar como era solicitado dos vectores de igual dimensión para así recibir la velocidad de mezcla, se prosiguió a programar uno mediante el uso de sumadores y compuertas lógicas.	Ver video demostrativo en la carpeta de nombre "comandar".	Asignando valores variables de entrada de ambos vectores a multiplicar y regularlos con los <i>switchs</i> de la Placa, se muestra el producto final en los LEDs, siendo los prendidos los valores lógicos 1.

<p>Procesamiento correcto del segundo byte obtenido correspondiente al byte de Comandos requerido.</p>	<p>El segundo byte era el encargado de definir la acción a realizar, tales como el fijar la temperatura y la velocidad de la mezcla, al igual que el indicar el inicio de un programa.</p>	<p>Ver video demostrativo en la carpeta de nombre “comandar”.</p>	<p>Con ayuda de un esquema de la asignación de <i>switchs</i>, botones y LEDs, se observa la correcta acción gráfica necesaria para cada Comando. De la misma forma, se ve claramente como a diferentes combinaciones de los <i>switchs</i>, también varía el byte de Datos de forma dependiente del Comando elegido.</p>
<p>Trato satisfactorio del tercer byte recibido correspondiente al byte de Datos precisado.</p>	<p>El tercer byte, variando dependiendo de cada Comando elegido en el byte anterior, poseía el deber de especificar las condiciones de la acción definida por los bytes anteriores.</p>	<p>Ver video demostrativo en la carpeta de nombre “comandar”.</p>	
<p>Modulación de la señal adecuada a la precisada en el caso de dividir la frecuencia a 10Hz (obteniendo un periodo de 100 ms).</p>	<p>De manera que el contador decimal pudiera contar las décimas de segundo y, por consiguiente, las posteriores unidades y decenas de segundo, este modulador era sumamente relevante para el trabajo.</p>	<p>Ver video demostrativo en la carpeta llamada “reloj-contador” en el archivo de nombre “contador decimal”.</p>	<p>Se observa claramente la cuenta de decenas en HEX2, unidades en HEX1 y décimas de segundo en HEX0, con el respectivo y solicitado punto decimal a continuación de la cifra de unidades, dentro del mismo <i>display</i>. El reloj modulado a 10 Hz se encuentra visible en uno de los LEDs de la Placa de Altera.</p>
<p>Cuenta correcta y temporalmente coherente del tiempo en décimas, unidades y decenas de segundo. La mayor cifra considerada necesitó ser reiniciada en el valor 6.</p>	<p>Despliegue adecuado de la cuenta de los contadores concatenados para su posterior uso en las pantallas o al momento de la cuenta de los programas, debiéndose regir esto por la cantidad de segundos</p>	<p>Ver video demostrativo en la carpeta de nombre “contador decimal”.</p>	

	transcurridos entre estados.		
Funcionamiento coherente del bloque “compararCuenta”.	Para la realización de la máquina de estados de los programas, era importante el conocer la cuenta actual y compararla con la necesaria para el cambio de estado. Así, el bloque mencionado era el encargado de comparar las tres cuentas recibidas del contador decimal (decenas, unidades y décimas de segundos).	Ver video demostrativo en el archivo de nombre “comparar cuenta”.	Mirando atentamente el video anterior, se visualiza como correctamente un LED se encienda cada vez que la cuenta es igual a 15 y 20 segundos, verificando la utilidad del bloque.
Funcionamiento correcto de los programas junto con sus correspondientes máquinas de estado.	Para la correcta implementación de los programas, se debió crear y programar una lógica de máquina de estados para cada uno, la cual se correspondiera con las pantallas de modo de desplegar información coherente.	Ver video demostrativo en el archivo de nombre “programa 2”.	Se visualiza como, en el caso del Programa 2 ejemplificado en el video anterior, al permanecer en un estado, enciende el o los LEDs correspondientes a la acción respectiva para ese estado.

<p>Procedimiento coherente y satisfactorio del bloque Macroprograma que engloba a todos los programas.</p>	<p>De modo de coordinar y agrupar a los 8 programas existentes y requeridos para el Obligatorio, se prosiguió a crear un nuevo bloque de mayor jerarquía.</p>	<p>Ver video demostrativo en el archivo de nombre “macroprograma”.</p>	<p>Variando con los <i>switchs</i> que programa se encuentra procesando la Placa, se observa una continuación de los procesos ocurridos y correspondencia respecto a la cuenta también desplegada.</p>
<p>Despliegue apropiado de las Animaciones solicitadas (llenado, girado horario y antihorario, y compresión.</p>	<p>Siguiendo el diseño de las animaciones solicitado por la letra del trabajo, se programaron las mismas con ayuda de un registro de desplazamiento.</p>	<p>Ver video demostrativo en la carpeta de nombre “animaciones”.</p>	<p>Se observa un <i>display</i> de cada animación en funcionamiento a una frecuencia de 10 Hz (periodo de 100 ms).</p>
<p>Muestra en los <i>displays</i> de la información según el formato de Pantalla 1.</p>	<p>En la modalidad de Pantalla 1, se solicita la animación del proceso actual en HEX3 junto a la cuenta decimal en HEX2, HEX1 y HEX0.</p>	<p>Ver video demostrativo en la carpeta llamada “pantallas”, en el archivo de nombre “pantalla 1”.</p>	<p>Variando con los <i>switchs</i> el proceso actual en acción, se prosigue a observar todas las animaciones solicitadas.</p>
<p>Creación y procesamiento satisfactorio del convertidor de bits a BCD.</p>	<p>De modo de hacer posible el despliegue de la velocidad y la temperatura de la mezcla en los LEDs requeridos, se precisó poder separar el byte recibido en 3 vectores de 4 bits.</p>	<p>Ver video demostrativo en el archivo de nombre “conversor BCD”.</p>	<p>Se visualiza como al variar el número ingresado en forma de byte (8 bits) con los <i>switchs</i>, varía también la cuenta en los <i>displays</i>.</p>

Despliegue en los <i>displays</i> de la información a modo del formato de Pantalla 2.	Continuando con lo especificado para la Pantalla 2, se debía expresar la velocidad de la mezcla en ese momento con su respectivo signo en caso de ser el sentido del giro el antihorario.	Ver video demostrativo en el archivo de nombre "conversor BCD". Se observa un funcionamiento análogo para la Pantalla 3.	Se testea controlando el sentido y la existencia del giro, junto con la velocidad a desplegar en la pantalla según el convertidor anterior.
Desarrollo en los <i>displays</i> de la información como el formato de Pantalla 3.	Según lo regulado para la Pantalla 3, una vez activada se necesitó expresar la temperatura de la mezcla en los tres primeros <i>displays</i> .	Ver video demostrativo en la carpeta llamada "pantallas", en el archivo de nombre "pantalla 3".	Similarmente al caso anterior, se muestra como se despliega la temperatura de la mezcla, regulada por los <i>switchs</i> y usando el convertidor.
Modulación satisfactoria de la señal de entrada a una frecuencia de 0.5 Hz correspondiente a 2 segundos.	De forma de facilitar la comparación de tiempos en la máquina de estados de la Pantalla 4, se crea un modulador de 2 segundos.	Ver video demostrativo en la carpeta llamada "reloj-contador", en el archivo de nombre "modulador 0.5Hz".	Se observa el apagado del LED cada 2 segundos, correspondiéndose con cuando la señal de reloj posee cada flanco ascendente.
Procesamiento en los <i>displays</i> de la información como el formato de Pantalla 4.	Verificación del funcionamiento de la Pantalla 4 con su respectiva máquina de estado, encargada de desplegar el porcentaje de llenado de cada válvula de entrada en cada estado correspondiente, debiendo	Ver video demostrativo en la carpeta llamada "pantallas", en el archivo de nombre "pantalla 4".	Variando el programa de entrada según los <i>switchs</i> , se observa el correcto <i>display</i> de los valores solicitados para el Programa 2 de ejemplo enseñado. Se observa que el modulador utilizado es de 2 Hz y por tanto, de 0.5

	variar la válvula enseñada cada 2 segundos.		segundos, corregido para la versión final.
Funcionamiento correcto y adecuado del despliegue del bloque Macro Pantalla.	Buscando la implementación de información en los <i>displays</i> según las 4 pantallas anteriores, se creó un nuevo bloque de mayor jerarquía.	Ver video demostrativo en el archivo de nombre "macropantalla".	Asignando a los <i>switchs</i> para que determinen el programa actual que se encuentra corriendo al igual que la pantalla desplegada, se presenta una demostración del funcionamiento del bloque.
Procesamiento Global de las mezcladoras de modo correcto, adecuado y manteniendo coherencia entre las partes anteriores.	Manteniendo una organización coherente y lógica entre lo enseñado en los <i>displays</i> y los programas transcurriendo en el trasfondo, este testeo refiere al funcionamiento de una mezcladora en general, repitiéndose para el caso de cada una de las cuatro mezcladoras a usar en el trabajo final.	Ver video demostrativo en el archivo de nombre "mezcladora".	Con los <i>switchs</i> , LEDs, botones y <i>displays</i> de 7 segmentos asignados de la manera detalladamente explicada por la letra del Obligatorio 2, se muestra el programa 2 ya configurado en la Placa FPGA (se asignaron valores de 255 y 254 a la velocidad y temperatura de la mezcla respectivamente).

Tabla 13. Tabla de Prueba indicando casos de prueba demostrativos estudiados y filmados o grabados para su posterior estudio y visualización

12. Diagrama de Tiempos

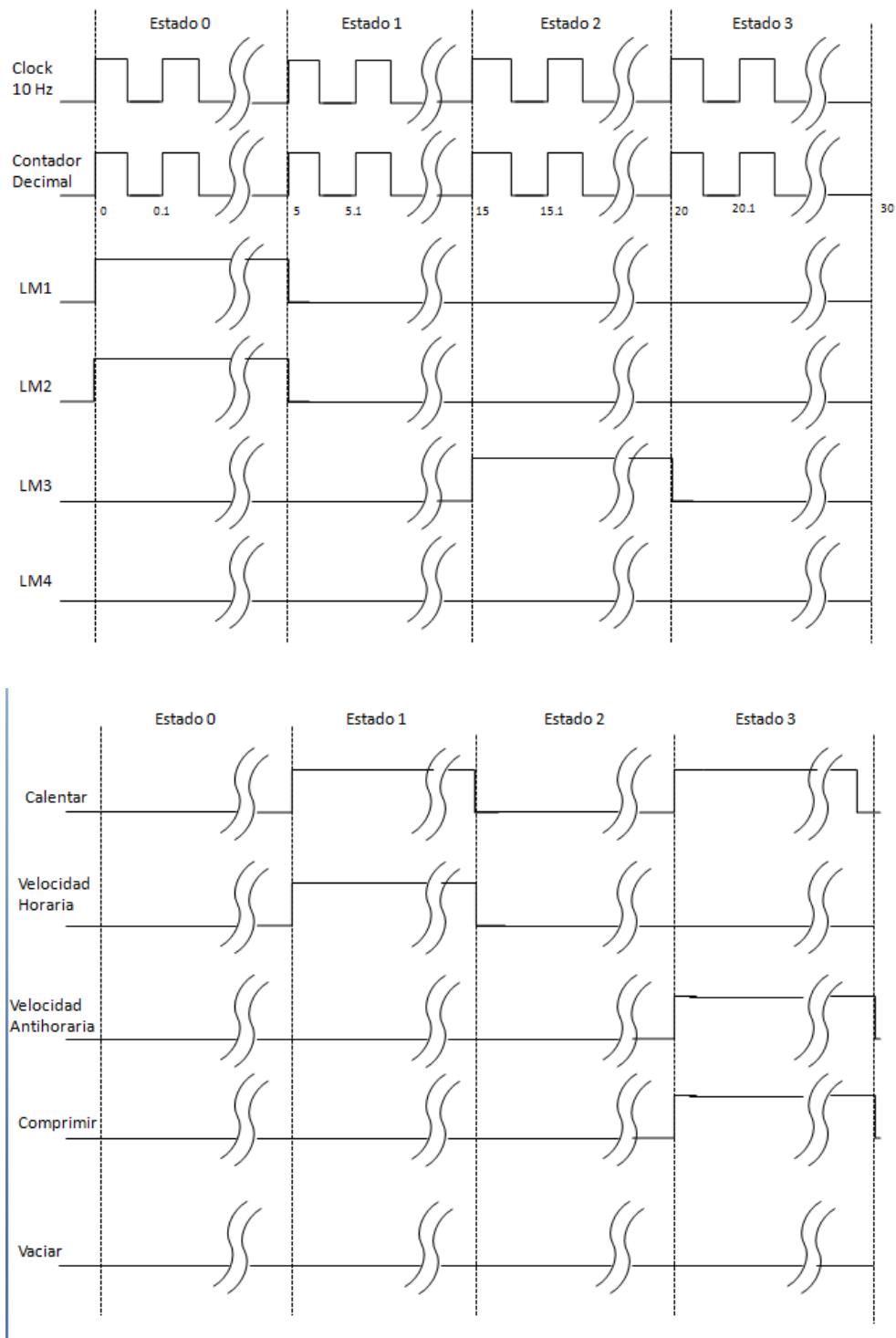


Figura 54. Diagrama de Tiempos de las principales señales referentes a los estados 0 a 3 del Programa 2

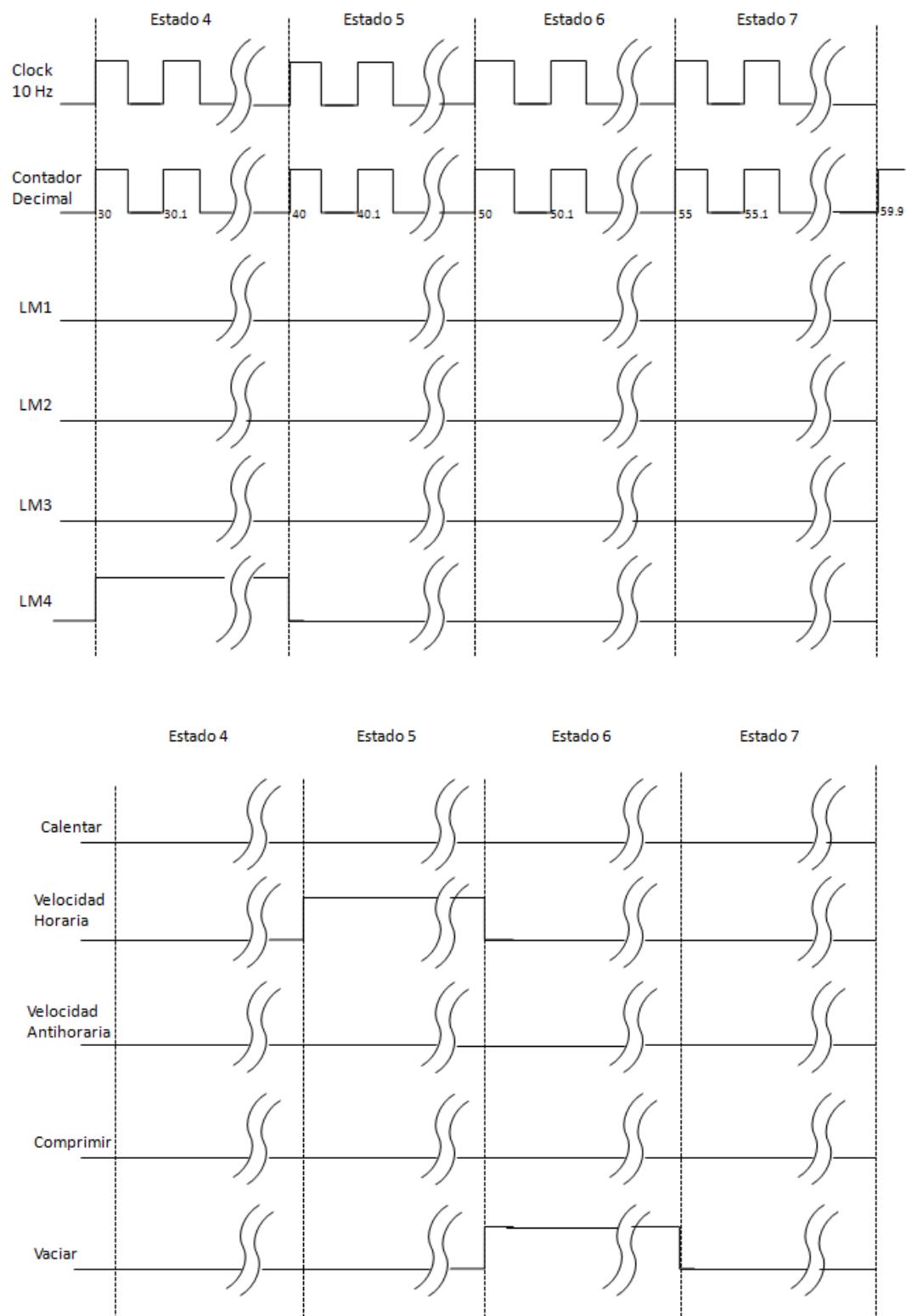


Figura 55. Diagrama de Tiempos de las principales señales referentes a los estados 4 a 7 del Programa 2

Referencias Bibliográficas

- . [1] Texas Instruments, “Quad 2 input NAND Gate.” (2015, Septiembre 17). [Online]. Disponible: http://pdf.datasheetcatalog.com/datasheets/208/108647_DS.pdf