BLOQUE TEMÁTICO 1

TÍTULO DE LA ACTIVIDAD: Diseño jerárquico de un sistema digital – Reloj Programable	**CÓDIGO: BT1_A1_P0**

FECHA:			
NOMBRE:		APELLIDOS:	

MODALIDAD:	Individual. Resolución de problemas	TIPO:		DURACIÓN:	17 horas
CALENDARIO:	Parte no presencial: PS3, 4 y 7 Parte presencial: sesiones S1, S2, S3 y S4	REQUISITOS:			
CRITERIO DE ÉXITO:					

COMENTARIOS E INCIDENCIAS:

TIEMPO DEDICADO:	minutos	AUTOEVALUACIÓN: [entre 0 y 10 puntos]	No procede

1. Introducción

Expresar la funcionalidad de un sistema digital complejo mediante un conjunto de subsistemas interconectados, consiguiendo una buena solución (económica y eficaz), obliga a emplear la creatividad del diseñador y también a aplicar múltiples conocimientos y habilidades técnicas:

- Es necesario conocer el funcionamiento e interfaz de los tipos de subsistemas habitualmente empleados en la realización de sistemas digitales complejos: multiplexores, codificadores, decodificadores, circuitos aritméticos, registros, contadores, temporizadores, monoestables, memorias (RAM, ROM, FIFOs), etcétera. Se deben conocer las distintas variantes de cada subsistema, sus aplicaciones típicas y se debe poder valorar su complejidad estructural, el coste en recursos hardware de su materialización y el esfuerzo que supone su modelado y depuración.

  Porque realizar el diseño jerárquico consiste esencialmente en expresar la funcionalidad de un sistema complejo como una estructura de subsistemas simples y conocidos por el diseñador.

- Es necesario dominar las técnicas de análisis del funcionamiento de sistemas digitales complejos.

  Para poder evaluar eficazmente la validez de las distintas soluciones que se le puedan ocurrir al diseñador para resolver la realización de funciones complejas, tarea que en primera instancia suele abordarse sin ayuda de herramientas software.

- Es necesario tener la capacidad de comparar las distintas soluciones que pueden resolver una determinada funcionalidad.

  Para elegir las alternativas que tienen menor coste hardware y requieren un menor esfuerzo de desarrollo.

En esta actividad se abordará el diseño jerárquico de un sistema digital complejo, con el objetivo de mostrar técnicas y procedimientos que se pueden aplicar en esta tarea que, de por sí, tiene un carácter escasamente sistematizable. La realización de la actividad se divide en cinco partes:

1. En la sesión presencial S1 se dedican 30 minutos a la descripción de las especificaciones del diseño “Medidor de Temperatura y Humedad MEDTH” (BT1_A1_P1)

2. En la sesión presencial S1 se dedican 45 minutos a explicar la metodología de diseño jerárquico y el diagrama de bloques de alto nivel del diseño (BT1_A1_P2).

3. En la sesión presencial S2 se dedica una hora a describir la funcionalidad e interfaz del bloque cuyo estudio en profundidad se aborda en esta actividad, el reloj programable (apartado 2) y a presentar un primer diagrama de bloques en el que se plasma una primera diferenciación funcional dentro del bloque (BT1_A1_P3). Posteriormente se dedica otra hora a leer los apartados 3 y 4 en los que se aborda el análisis de la funcionalidad de los bloques para determinar en cuáles de ellos resulta necesario profundizar en la división funcional y se lleva a cabo la depuración de este diseño preliminar (BT1_A1_P3).

4. En la sesión no presencial previa a la sesión S3 se dispone de una hora para leer el apartado 5, dedicado a la elaboración de la documentación definitiva del diseño jerárquico.

5. En la sesión presencial S3 se dedican 50 minutos a la realización del apartado 6 y a la resolución de dudas relacionadas con el diseño lógico del bloque del reloj programable. Se dedicarán los 50 minutos restantes a explicar las funciones, procedimientos y paquetes en el lenguaje VHDL y la sentencia *assert*, y su aplicación en el desarrollo de *test-benches* para la simulación y verificación de sistemas complejos (BT1_A1_P4).

6. En la sesión no presencial previa a la sesión 4 se realizará parte del apartado 7. En concreto, se llevarán a cabo los apartados a, b, c y d del ejercicio 5, en los que se muestra un ejemplo de uso de estas herramientas del lenguaje para la detección de un error en el modelado del reloj programable.

7. En la sesión presencial S4 se dedicarán 100 minutos a la finalización del apartado 7. Concretamente, se realizarán los apartados finales del ejercicio 5: e y f.

8. En sesiones no presenciales posteriores se dedican seis horas al apartado 8 en el que se debe realizar la depuración del resto de los errores cometidos en el diseño del bloque; esta tarea debe completarse antes de la sesión presencial S7.

Al final de este documento se recogen soluciones a algunos de los ejercicios planteados en la actividad. Debe revisarlas sólo después de haber realizado dichos ejercicios.

2. Especificación de la funcionalidad del sistema

Se trata de realizar el bloque del diseño del "Medidor de Temperatura y Humedad MEDTH" correspondiente al reloj. El reloj dispone de visualización de horas, minutos y segundos, y permite establecer la hora mediante distintos procedimientos. Por tratarse de un bloque que se encuentra insertado en un sistema de mayor complejidad, las especificaciones de su funcionamiento e interfaz habrán venido determinadas por un análisis de orden superior, esto es, por el análisis de la función que debe desempeñar este bloque en el conjunto de la jerarquía y su interrelación con el resto de bloques de la misma. A los efectos del desarrollo de las actividades del BT1, ha de considerar que este análisis se ha realizado, habiéndose identificado la interfaz y función de los bloques funcionales del nivel superior de la jerarquía del diseño. Esta partición se ha documentado en la memoria del diseño, que deberá leer durante la realización de la Tarea 1. En la primera sesión presencial, además, se le explicó un diagrama de bloques simplificado de este primer nivel de la jerarquía del diseño.

A continuación se muestra un resumen de la parte del análisis correspondiente a la interfaz y función del bloque RELOJ. La interfaz del bloque RELOJ toma como punto de partida las especificaciones globales del sistema con relación al funcionamiento del reloj. En este sentido, sabemos que se pretende realizar un reloj con una resolución de un segundo y, tras la división preliminar de funcionalidades, también sabemos que el sistema dispone de un bloque temporizador que proporciona un conjunto de señales a modo de "tics". Así pues, uno de estos tics deberá tener un periodo de un segundo y estar conectado a una entrada de nuestro bloque creada para tal propósito (TIC_1S). Por otra parte, la programación y control del reloj se realiza mediante un teclado hexadecimal cuyo bloque de control identifica la tecla pulsada con un código binario de 4 bits, distinguiendo además entre pulsaciones cortas (de menos de 2 segundos) o largas (de 2 segundos o más). Teniendo esto en cuenta, el bloque de reloj incluirá una entrada que identifique la tecla pulsada (CMD_TECLA), con cuatro bits, puesto que se trata de un teclado de 16 teclas; y entradas para identificar pulsación breve (ENA_CMD) o larga (PULSO_LARGO). Con relación a la interfaz de salida, el reloj debe generar la información correspondiente a los segundos, minutos y horas. Por motivos de simplicidad en el manejo de esta información por parte de los bloques que hagan uso de ella, parece razonable dar esta salida en código BCD. Por consiguiente, forman parte de la interfaz las señales SEGUNDOS, MINUTOS y HORAS, todas ellas de 8 bits con el fin de que puedan albergar dos dígitos BCD cada una. A esto se añaden dos señales más que ayudan a interpretar la información que se obtiene por las señales anteriores, ya que el reloj debe proporcionar la hora en dos posibles formatos, 12 horas o 24 horas. De esto se encargan las señales MODO y AM_PM. Finalmente, se ha decidido dotar al reloj de una salida, INFO, que identifique el campo que está siendo programado, en su caso. Esta información será utilizada el bloque encargado de la visualización para hacer parpadear el dígito que se esté programando. La figura 1 se muestra el conjunto de señales que constituyen la interfaz del bloque RELOJ:

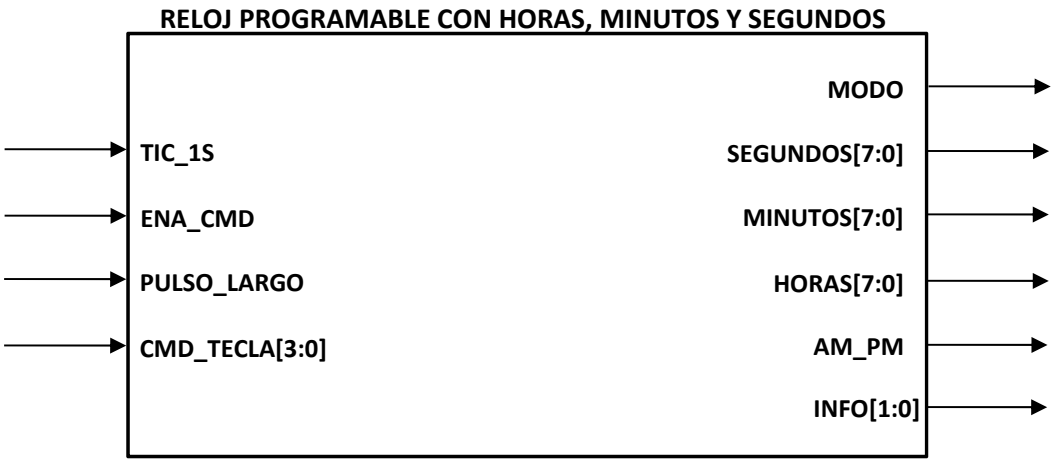


Figura 1

Durante la fase de diseño del nivel superior de la jerarquía del diseño, se ha determinado que la funcionalidad del bloque RELOJ debe ser la siguiente:

- El reloj ofrecerá en sus salidas los **segundos**, **minutos** y **horas** de la hora actual expresada en código **BCD**.

- Dispone de **modos de representación**, en **12h** o en **24h**. Con la salida **modo** informa de cuál de los dos modos está activo. La salida **AM_PM** indicará el periodo actual.

- Los **comandos de control** se introducirán mediante un teclado hexadecimal, en el que se distinguirá entre pulsaciones breves y pulsaciones largas. La señal **cmd_tecla** contendrá el valor hexadecimal de la tecla pulsada, y las señales **ena_cmd** y **pulso_largo** se activarán con una pulsación breve o larga respectivamente.

- En el **modo normal** el reloj presentará la hora con una resolución de segundos. En el **modo de programación** se permitirá la edición de los minutos y de las horas, no así el campo correspondiente a los segundos que tomará de manera automática el valor cero una vez acabada la edición.

- La **entrada** al modo programación se realizará mediante la pulsación **larga** de la **tecla 'A'**. Siempre se entra en el modo de edición de horas.

- La **salida** del modo de programación se llevará a cabo mediante la pulsación **breve** de la **tecla 'A'**, o bien tras siete segundos de inactividad en el teclado.

- En el modo de programación la **tecla 'B'** permitirá **cambiar el campo** en edición, alternando entre minutos y horas a cada pulsación.

- La edición del campo que se encuentre seleccionado se podrá efectuar de varias formas. Se podrá **incrementar** en la unidad el **valor** de un **campo** mediante la pulsación breve de la **tecla 'C'**. Sin embargo, si se lleva a cabo una pulsación **larga** de la **tecla 'C'**, el valor del

campo se incrementará **cuatro veces por segundo** mientras esta tecla se mantenga activada. Igualmente podremos **introducir un valor** directamente con el teclado.

- Sea cual sea el modo del reloj se podrá **cambiar el formato 12h y 24h** mediante la pulsación breve de la **tecla 'D'**.

- La salida "**info**" indicará qué campo está siendo editado, o si el reloj se encuentra en modo de funcionamiento normal, presentando la hora.

Finalmente, se muestra una descripción más detallada y organizada de las señales de entrada y de salida que forman la interfaz del bloque.

Entradas:

- TIC_01S: Señal con una frecuencia de 1 Hz, en la que el nivel alto dura un ciclo de reloj.
- ENA_CMD: señal de un bit que informa de la pulsación breve de una tecla en el teclado hexadecimal, mediante un pulso a nivel alto de un ciclo de reloj de duración.
- PULSO_LARGO: señal de un bit que informa de la pulsación larga de una tecla del teclado, permanece a nivel alto mientras el usuario mantenga la tecla pulsada.
- CMD_TECLA: señal de cuatro bits que contiene el código hexadecimal que identifica la tecla pulsada, validado con las dos señales anteriores.

Salidas:

- SEGUNDOS: salida de 8 bits que contiene los dos dígitos BCD que contienen los segundos de la hora actual.
- MINUTOS: señal de 8 bits que contiene los dos dígitos BCD que contienen los minutos de la hora actual.
- HORAS: salida de 8 bits que contiene los dos dígitos BCD que contienen las horas de la hora actual.
- MODO: señal de un bit que indica si el reloj está en modo 12h(con un 0) o en modo 24h (con un 1)
- AM_PM: señal de un bit utilizada en el modo 12h y que indica, con un cero, el periodo AM, y con un 1, el periodo PM.
- INFO: señal de 2 bits que informa si algún dígito está siendo editado o si no lo está siendo ninguno. Para ello utiliza el código siguiente: "00" ninguno, "01" minutos y "10" horas.

3. Propuesta inicial de diagrama de bloques de la jerarquía.

A partir de este momento se va a mostrar el proceso de diseño de la jerarquía dentro del bloque de reloj. El proceso es el mismo que se utilizó en su momento para el diseño del nivel superior de la jerarquía del diseño, pero focalizado en el bloque de reloj en lugar de en el diseño completo.

En la interfaz anteriormente expuesta no figuran, por simplicidad, dos señales esenciales en todo sistema secuencial síncrono, una señal de reloj y una señal de reset asíncrono. Puesto que se presupone que se va a realizar un diseño síncrono con un solo dominio de reloj, se precisará únicamente de una señal de reloj y una de reset asíncrono, que, deberán ser las mismas que se utiliza como señales de reloj y reset del sistema.

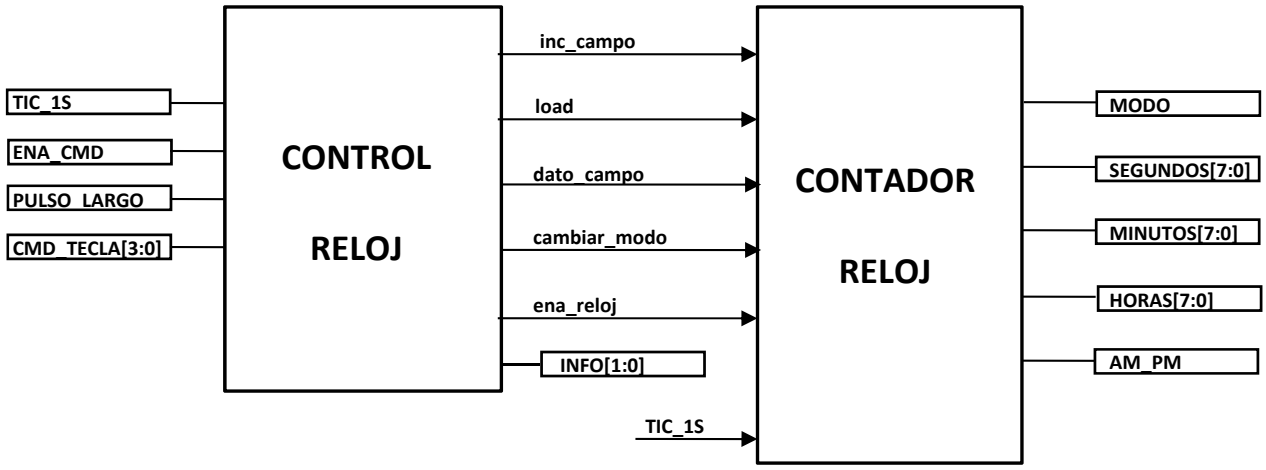
Para empezar a progresar en el diseño jerárquico preliminar hay que analizar la funcionalidad global que se espera del sistema y realizar, si se considera oportuno, una primera partición en bloques de esta funcionalidad que siempre será tentativa, teniendo en cuenta que se está en una fase de "exploración" de la estructura del diseño y, de momento, todo es provisional.

La funcionalidad global del sistema es la implementación de un reloj, por lo que una parte claramente diferenciada en esta funcionalidad es el reloj propiamente dicho, que deberá estar formado por un conjunto de contadores enlazados, que lleven la cuenta de los segundos, minutos y horas que van transcurriendo, bloque *Contador Reloj* (CNT_RELOJ). Por otra parte, existe un conjunto de funcionalidades asociadas a la interpretación de los comandos recibidos por el teclado a partir del cual habrá que hacer evolucionar al reloj por distintos estados y habrá que generar el conjunto de señales necesario para controlar aspectos tales como la modificación de la hora que está siendo programada o el modo de visualización de la hora, bloque *Controlador Reloj* (CTRL_RELOJ).

En la figura 2 se muestra una versión inicial del diagrama de bloques, en la que, partiendo de la especificación la funcionalidad global del sistema, se deduce, y describe someramente, la función o funciones de cada bloque y las señales que los conectan.

El diagrama de bloques de la figura 2 representa una partición viable de la funcionalidad del sistema. El siguiente paso para avanzar en un diseño preliminar de la estructura jerárquica del sistema es analizar si resulta necesario, o no, profundizar en la descomposición de los dos bloques que componen este nivel de la jerarquía.

Comenzando por el bloque correspondiente al **controlador del reloj** (CTRL_RELOJ) en un primer análisis deberá estar formado por un contador que realice la medida del tiempo de inactividad del teclado (timeout), un registro, o conjunto de registros, donde almacenar la información que se va recibiendo por el teclado; un autómata que controle la evolución por los distintos estados en los que se encuentra el reloj, en función de los comandos que van llegando (edición de minutos, horas, modo normal, etc.); y una lógica combinacional para la interpretación de los comandos y para la generación de las señales del control del bloque CNT_RELOJ. En principio, parece una tarea abordable en un solo bloque y, salvo replanteamiento posterior, no consideramos una subdivisión adicional de este bloque.



- INTERPRETA EL COMANDO INTRODUCIDO POR CMD_TECLA PARA DETERMINAR LA ACCIÓN A REALIZAR.
- GENERA LAS SEÑALES QUE CONTROLAN LA MODIFICACIÓN DE LA HORA EN EL MODO DE PROGRAMACIÓN.
- HABILITA EL RELOJ EN EL MODO NORMAL.
- INFORMA DEL ESTADO ACTUAL DEL RELOJ.

- REALIZA LA CUENTA DE SEGUNDOS, MINUTOS Y HORAS.
- MODIFICA LOS CAMPOS EN EL MODO PROGRAMACIÓN.
- COMPONE LA SALIDA EN FORMATO 12h O 24h.
- GENERA TODA LA INFORMACIÓN SALIDA DEL RELOJ

Figura 2

Mención aparte merecen los detalles de este conjunto de señales de control. Estas son: *inc_campo*, señal que indica el campo que ha de ser incrementado; *load*, señal que indica qué campo ha de ser cargado en paralelo; *dato_campo*, señal que contiene el valor a cargar en el campo correspondiente; *cambiar_modo*, señal que ordena el cambio de modo; *ena_reloj*, señal que habilita la cuenta de tiempo en el modo normal.

Con relación al bloque **contador de reloj** (CNT_RELOJ) en una primera aproximación se detecta una modularidad inherente, puesto que debemos llevar la cuenta de los segundos, los minutos y las horas, para lo cual necesitaremos modelar sendos contadores BCD para unidades y decenas, en cada uno de los tres casos. A priori no parece una funcionalidad que no pudiera ser implementada en un solo bloque sin una ulterior subdivisión en bloques más simples. Sin embargo, hemos de tener en cuenta que además del incremento cada segundo debido al funcionamiento normal de un reloj, debemos manejar también la carga o incremento de cada uno de los campos. Esta reflexión nos lleva a la conclusión de que podría facilitar las labores de diseño centrarnos en el control de cada uno de los campos de forma separada de los demás. En consecuencia, se propone la división de este bloque en el diagrama mostrado en la figura 3.

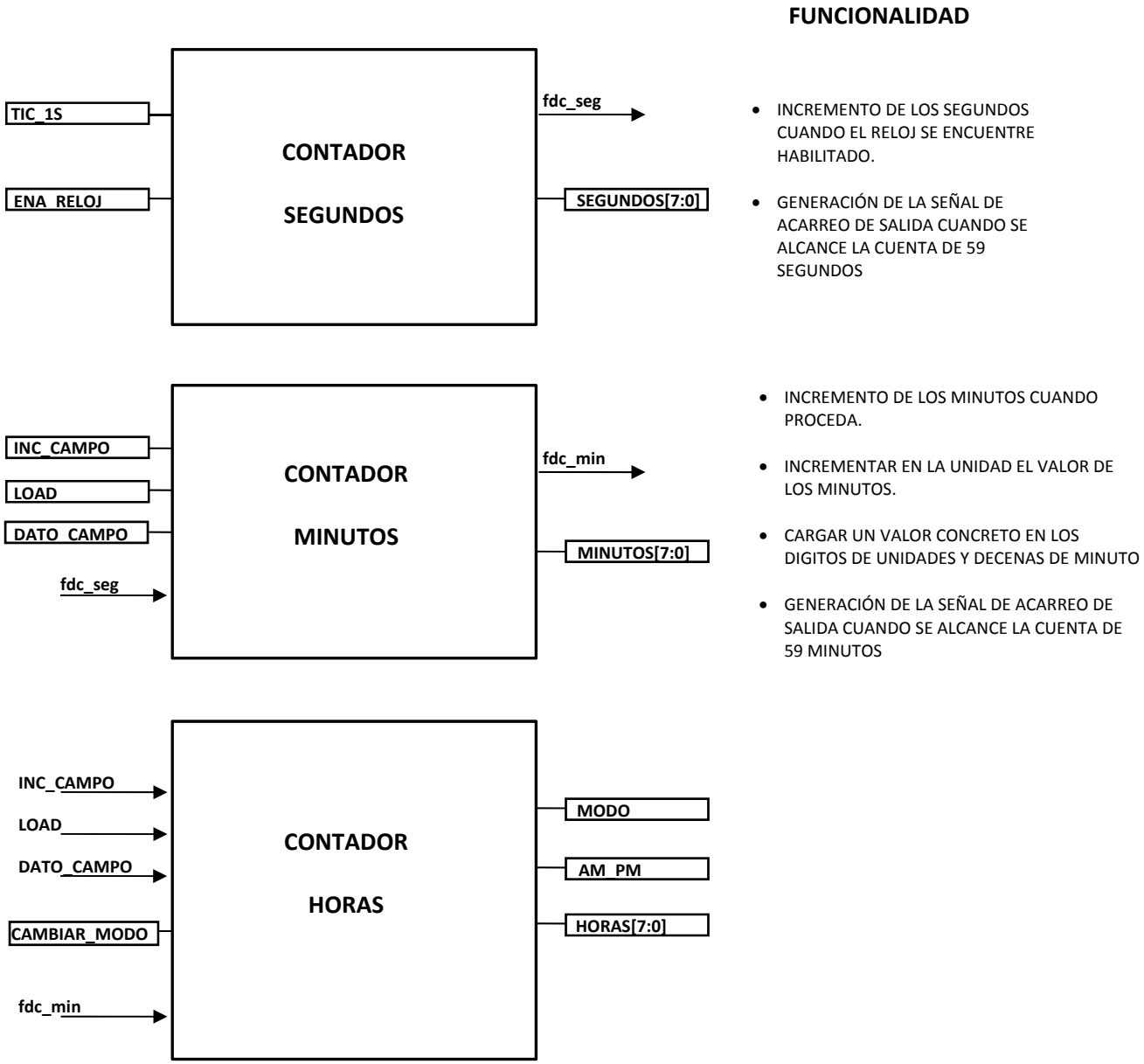


Figura 3

Ejercicio 1

A partir de la descripción de la funcionalidad completa que debe tener el bloque de reloj, proponga el conjunto de funcionalidades que, a su juicio, deberá desempeñar el bloque CNT_HORAS.

4. Depuración del diseño preliminar

A partir del momento en que se descarta continuar descomponiendo módulos, se dispone de una versión de partida (preliminar y poco detallada) sobre la que se puede trabajar para obtener una realización satisfactoria del diseño jerárquico del sistema. Lo primero que hay que hacer con ella es validarla funcionalmente, es decir, comprobar que es capaz de cumplir con todos los requisitos funcionales el sistema. La realización de esta tarea permitirá, además de depurar errores que se hayan cometido en el diseño preliminar, ir detallando las funciones y las características de las interfaces de los bloques.

Para validar funcionalmente el diseño jerárquico hay que hacer, primero, una lista de todas las operaciones que debe ser capaz de realizar el sistema y, después, comprobar, para cada una de ellas, cómo puede ser realizada por los bloques en que se ha dividido el diseño, estando conectados tal y como se ha determinado en la versión preliminar.

Un ejemplo de una operación que debe realizar el sistema es:

1. Entrar en el modo de programación de la hora actual cuando se efectúa una pulsación larga de la tecla ‘A’ y el reloj está funcionando en el modo normal.

Una descripción de cómo el diseño jerárquico preliminar podría ejecutar esta operación sería:

*“Se activa la señal **pulso_largo** y en **cmd_tecla** aparece el código ‘0xA’ lo cual será un indicativo de una pulsación larga en la tecla A por parte del usuario. Esta información es procesada por el bloque **CTRL_RELOJ** el cual modifica el estado del **autómata** para **pasar** del modo normal al **modo programación**, concretamente a la edición de las horas. Igualmente, desactivará, con un cero, la señal **ena_reloj**, para evitar que el reloj siga avanzando el tiempo, e informará del cambio de estado modificando el código ofrecido por su salida **info**. El bloque **CNT_RELOJ** detectará el nivel bajo en la señal **ena_reloj** y **detendrá los contadores** decimales correspondientes a segundos, minutos y horas.”*

La descripción de cómo se realiza la operación permite:

- **Comprobar que existen los recursos necesarios para completarla**: Autómata de control, intérprete de códigos de tecla pulsada, señales para controlar la puesta en marcha y parada de los contadores de los que está constituido el reloj.

- **Definir detalles de las señales de las interfaces de los bloques**: activación de pulso_largo, ena_cmd no se activa, ancho de bits de cmd_tecla, y codificación utilizada, hexadecimal. Salida de habilitación de reloj activa a nivel alto y selección de códigos para la salida *info.*

- **Refinar la descripción de la funcionalidad de los bloques**: De la descripción de la operación se deriva que una parte esencial del bloque de control es un autómata que distingue el estado en el que se encuentra el reloj, estados por los que irá evolucionando según la tecla que se vaya pulsando. Otra parte importante de este bloque será la lógica destinada a interpretar el código de tecla pulsado. También tomamos conciencia en este momento la importancia de una correcta planificación de operaciones a realizar para ejecutar la acción solicitada por el usuario, ya que alguna de ellas puede ser compleja, por ejemplo, la edición de los campos del reloj mediante valores introducidos por el teclado.

- **Detectar errores en el diseño preliminar:** realizar el ejercicio mental de plasmar las acciones que debe llevar a cabo el bloque que estamos diseñando en operaciones concretas entre señales, puede ayudarnos a tomar conciencia de la comisión de algún error de diseño. Podemos apercibirnos de la necesidad de una señal adicional no contemplada o alguna conexión entre bloques inicialmente no prevista. No es el caso en esta primera operación, pero tendremos ocasión de encontrar algún ejemplo enseguida.

La operación que acabamos de describir tiene dos operaciones complementarias a ella, se trata de:

2. Salir del modo de programación cuando se realiza una pulsación breve de la tecla 'A'.
3. Salir del modo de programación cuando se mantiene el teclado inactivo durante más de siete segundos.

Ejercicio 2

Describa, tomando como modelo la descripción de la operación 1, cómo se realizaría la operación número 2.

Nota: No es necesario volver a describir en detalle características o procedimientos idénticos a otros ya descritos para operaciones análogas.

La descripción de cómo se ejecuta la operación 3 podría resumirse de la siguiente manera:

*"El bloque **CTRL_RELOJ** dispondrá de un **contador** que irá contando tics de la señal de 1 segundo, siempre que el reloj se encuentre en el modo de programación. Este contador será inicializado siempre que se active la señal **ena_cmd** o **pulso_largo**, lo cual será un indicativo de una pulsación de tecla por parte del usuario. Cuando este contador alcance el valor umbral 7 tendremos la seguridad de que han trascurrido los **siete segundos**. En ese momento el autómata pasará al estado normal, lo cual será advertido mediante el cambio en la señal **info**. Por otra parte, deberá garantizarse que la señal **ena_reloj** se pone a nivel alto para permitir el funcionamiento del reloj con normalidad."*

4. Cambiar el campo que está siendo editado, con la tecla 'B'.

*"Cuando en el bloque **CTRL_RELOJ** la señal **ena_cmd** se activa, indicando un pulsación breve, y se identifica el código de tecla 0xB en la señal **cmd_tecla**, el autómata irá alternando entre los estados que representan la edición de los minutos y las horas. Por lo que cualquier orden subsiguiente de modificación de la información se aplicará sobre el campo que esté activo en edición en ese momento. El nuevo estado será notificado mediante el cambio en la señal **info**."*

5. Intercambiar entre el modo 12h y 24h, con pulsación breve de la tecla 'D'.

*"Cuando en el bloque **CRTL_RELOJ** se detecta una pulsación de tecla, **ena_cmd** activo, y el código detectado es el 0xD, este bloque activará con un nivel alto la señal **cambiar_modo** durante un ciclo de reloj. Esta señal está conectada a la entrada homónima en el bloque **CNT_RELOJ**, el cual al detectar esta activación recupera la información del **flip-flop** que almacena el modo actual y*

*lo invierte. En función del modo de funcionamiento activo la hora que será presentada llevará un formato u otro, por lo que habrá una **lógica combinacional** en el sistema que lleve a cabo la **corrección de la hora** en formato 24h, a partir de la hora en formato 12h y la señal **AM_PM**, y viceversa."*

6. Incrementar el valor de un campo, mediante pulsación breve en 'C'.

*"Tras la activación de la señal **ena_cmd** y la aparición del código 0xC en la señal **cmd_tecla**, el bloque CTRL_RELOJ activará, durante un ciclo de reloj, el bit de la señal **inc_campo** correspondiente al campo actualmente en edición. Esta señal deberá tener, por lo tanto **2 bits**, uno para cada uno de los campos, minutos y horas, que pueden ser objeto de edición. El bloque **CNT_RELOJ** recibe esta información y será el encargado de incrementar el **contador BCD** que almacena el campo correspondiente"*

7. Incrementar continuamente el valor de un campo, mediante pulsación larga en 'C'.

*"Tras la activación de la señal **pulso_largo** y la aparición del código 0xC en la señal **cmd_tecla**, el bloque CTRL_RELOJ activará, durante un ciclo de reloj cada 250ms, el bit de la señal **inc_campo** correspondiente al campo que se encuentre siendo editado (minutos u horas). El bloque **CNT_RELOJ** recibe esta información y será el encargado de incrementar los **contadores BCD** que almacenan los dígitos correspondientes al campo seleccionado. Dicho esto, sin embargo, nos encontramos con el problema de tener que efectuar una acción cada 250ms, cuando el tic que disponemos es de 1s. Tomamos pues conciencia de un error en el planteamiento inicial de la interfaz y de la necesidad de disponer de un tic de 250 ms (que llamaremos **tic_025s)** como una **entrada adicional del bloque CTRL_RELOJ**"*

8. Introducción directa con el teclado numérico de un nuevo valor para el campo.

*"Cuando el bloque **CRTL_RELOJ** detecta una pulsación de tecla, **ena_cmd** activo, y el código detectado es un número entre 0 y 9, se entiende que el usuario desea introducir directamente el valor de un campo. Este valor numérico será almacenado en un **registro**, ya que una activación posterior de **ena_cmd** con el código de otra tecla numérica, nos dará un nuevo número que deberá ser añadido junto al valor almacenado en este registro para conformar el valor de unidades y decenas del campo cuyo valor deseamos establecer. Asimismo, cada pulsación de tecla numérica provocará la activación del bit de la señal **load** que indica el campo destinatario del nuevo valor, y la entrega del nuevo valor a través de los ocho bits de la señal **dato_campo**. En el caso de la primera pulsación el valor a enviar deberá ser concatenado con el valor inicial del registro, que deberá ser cero. El bloque **CNT_RELOJ**, recibirá la indicación de la modificación del campo que se desean establecer mediante los bits de la señal **load**, por lo tanto, cuando se active alguno de estos bits, este bloque procederá a la **carga paralelo** del contenido de su señal de entrada **dato_campo** en el **contador BCD** adecuado.*

*Hay que considerar en este punto que este modo de operar puede ocasionar un mal funcionamiento del sistema. El problema radica en la posibilidad de que se trate de configurar de esta manera una hora que no es posible. Por ejemplo, se introduce un valor superior a 59 en el campo de los minutos; o, en el modo 12h, tratamos de introducir un valor en el campo de las horas superior a 11; y así podríamos describir unas cuantas posibilidades más. Así pues, en caso de introducirse un valor inválido el bloque CTRL_RELOJ deberá detectarlo y no activar la señal **load**. Esto conlleva la introducción de una lógica de control de errores que precisa del conocimiento del modo actual. Por*

*lo que será necesario realizar alguna **modificación en el diagrama de bloques preliminar**, concretamente la introducción de la señal **modo** como entrada del bloque **CTRL_RELOJ**"*

Respecto al funcionamiento normal de medida horaria del reloj podemos describir el siguiente conjunto de operaciones:

9. Incrementar la hora con una resolución de un segundo.

*"El bloque **CTRL_RELOJ** activará su salida **ena_reloj** para habilitar la cuenta en el bloque CNT_RELOJ. En el bloque **CNT_RELOJ** la señal **tic_1s** habilitará la cuenta de los segundos durante un ciclo de reloj cada segundo. El bloque lleva el control, internamente, de los acarreos entre los diversos dígitos de los segundos, minutos y horas. Igualmente, deberá incrementar de forma correcta para el campo de la hora según se encuentre en los modos 12h ó 24h. El valor de cada uno de los tres campos se ofrece en sendas salidas de 8 bits, utilizando para ello dos dígitos BCD en cada una."*

10. Realizar el paso de AM a PM, y viceversa, cuando el reloj está en modo 12h y se alcanza la hora 11:59:59.

*"Cuando el reloj se encuentra configurado en el modo **12h**, el sistema deberá recordar el periodo horario actual, **AM** o **PM**, y en el momento que se alcance el tiempo 11h:59m:59s, el siguiente estado de cuenta será el 00h:00m:00s y el indicador permutará de valor, cambiando consiguientemente la salida. En el modo **24h**, aunque su contenido podría ser irrelevante al no ser representado en los displays, parece prudente optar por mantener un valor consistente en esta señal con relación al modo 12h."*

Como resultado del análisis de las operaciones se ha refinado la estructura del diagrama de bloques de la figura 2, que quedaría tal y como se muestra en la figura 4.

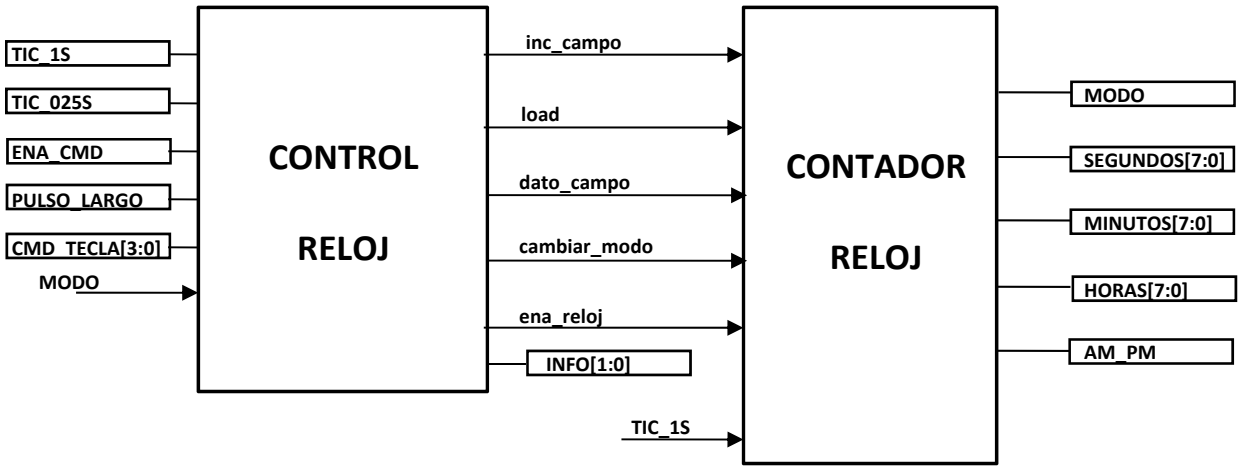


Figura 4

Otros productos aprovechables de los análisis realizados en la depuración del diseño jerárquico preliminar son:

- Una descripción más detallada de la forma y actividad de las señales de las interfaces de los bloques, que será de utilidad a la hora de realizar la versión detallada final del diseño jerárquico del sistema.

- Una descripción más detallada de las funciones de los bloques, que será de utilidad tanto a la hora de realizar la versión detallada final del diseño jerárquico, como al abordar el diseño lógico de los bloques.

- Una lista pormenorizada de las operaciones (funciones) que debe ser capaz de realizar el sistema; esta lista se **debe** utilizar como punto de partida para el diseño del *test plan* que debe realizarse para validar el sistema.

5. Realización de la documentación definitiva del diseño jerárquico del sistema

Para terminar, hay que realizar una versión detallada del diseño jerárquico en la que:

- Se defina la estructura definitiva de los distintos niveles de la jerarquía del sistema: se documentará mediante diagramas de bloques.

- Se defina en detalle para cada bloque:
  - ✓ la interfaz, detallando el nombre de las señales de entrada y salida e indicando, para cada una de ellas, la información que codifican, el código y número de bits empleado para codificarla, así como cualquier otra información sobre las características de la señal que pueda resultar relevante.

  - ✓ la función, especificada de manera completa y sin ambigüedades para cada bloque, mediante la descripción de la relación funcional de entradas y salidas.

La figura 5 representa el diagrama de bloques del nivel más alto de la jerarquía:

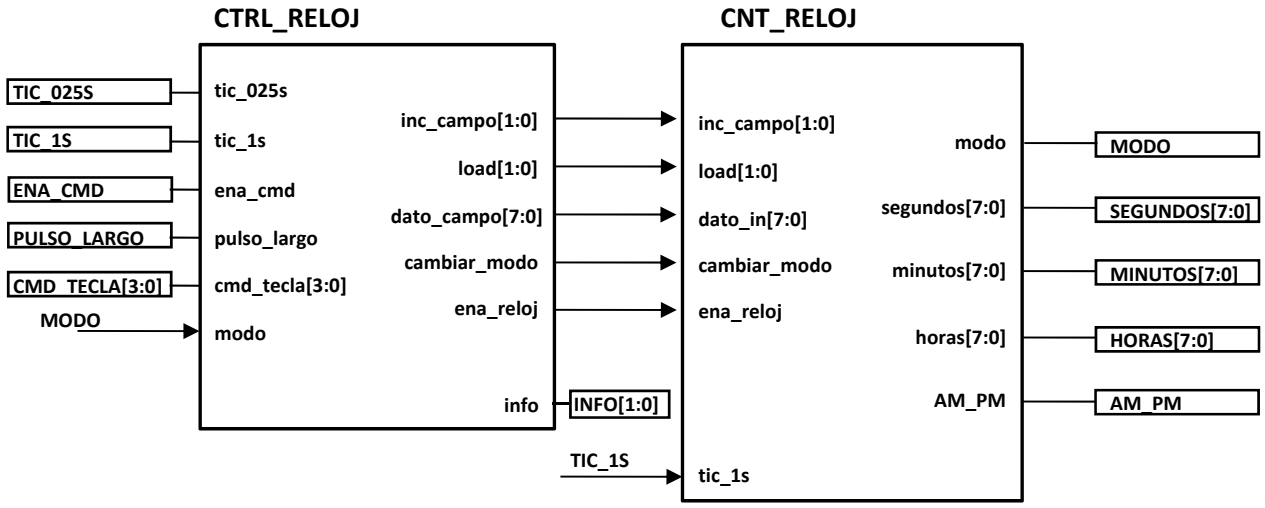


Figura 5

Y la descripción de la interfaz y funcionalidad de los bloques podría documentarse tal y como se muestra a continuación:

Bloque CTRL_RELOJ

Entradas:

- TIC_025S: Señal con una frecuencia de 4 Hz, en la que el nivel alto dura un ciclo de reloj.

El resto de señales de entrada: TIC_1S, ENA_CMD, PULSO_LARGO, CMD_TECLA y MODO; han sido ya descritas en la interfaz del Bloque Global de Reloj en el apartado 2.

Salidas:

- INC_CAMPO: señal de 2 bits que indica con un nivel alto qué campo debe ser incrementado, siguiendo el siguiente esquema: bit0, minutos; y bit1, horas.
- LOAD: señal de 2 bits que indica con un nivel alto qué campo debe ser modificado con el contenido de la salida "dato_campo", siguiendo el siguiente patrón: bit0, minutos; bit1, horas.
- DATO_CAMPO: dato a cargar en el campo señalado por la señal load. Los bits más bajos serán el valor a introducir en las unidades, mientras que el valor para las decenas se entregará en los cuatro bits más altos.
- CAMBIAR_MODO: se activará, con un nivel alto de un ciclo de reloj de duración, cuando se desee ordenar un cambio en el modo de operación del reloj. Mientras se encuentre a nivel bajo el modo permanecerá sin cambios.
- ENA_RELOJ: esta señal se mantendrá activa, a nivel alto, mientras el reloj deba estar funcionando con normalidad, es decir, incrementando su tiempo con una resolución de un segundo. Por consiguiente, mientras se encuentre a nivel bajo el reloj permanecerá parado.
- INFO: señal descrita en el apartado 2 en la descripción del bloque global de reloj.

Función: Este bloque está encargado de medir un tiempo de inactividad (sin activación de ENA_CMD o PULSO_LARGO) superior a siete segundos. Asimismo, es capaz de almacenar el último número (0 a 9) recibido en CMD_TECLA. Lleva el control del modo en el que se encuentra el reloj y activará sus salidas en función del comando recibido en CMD_TECLA, de la siguiente manera:

- La pulsación de una de las teclas será detectada mediante las señales CMD_TECLA, que contendrá el valor hexadecimal de la tecla pulsada, y las señales ENA_CMD y PULSO_LARGO que validarán este comando y que indicarán una pulsación breve o larga de la tecla en cuestión, respectivamente. La señal ENA_CMD deberá tener una duración de un pulso de reloj y la señal PULSO_LARGO deberá permanecer activa mientras se encuentre pulsada la tecla.
- Se pasará del modo "normal" al modo "programación" cuando se reciba el código de tecla '0xA' y la señal PULSO_LARGO se encuentre activada.
- El paso del modo de programación al modo normal se llevará a cabo cuando se reciba el código de tecla '0xA' y la señal ENA_CMD se encuentre activada.
- Al entrar en el modo de programación se habilita la posibilidad de cambiar las horas. La recepción del código de tecla '0xB', con la señal ENA_CMD a nivel alto, permitirá alternar entre el estado de edición de los minutos y de las horas, o viceversa.

- Si se recibe el código de tecla ‘0xC’ y la señal ENA_CMD está a nivel alto estando en el modo de edición de cada uno de los campos, se activará el bit de la señal INC_CAMPO que indica el campo que se tiene que incrementar.
- Estando en el modo de edición de cada uno de los campos y recibiendo el código de la tecla ‘0xC’ con la señal PULSO_LARGO activada, se activará el bit de la señal INC_CAMPO que indica el campo que se tiene que incrementar, siempre que la señal TIC_025s esté a nivel alto.
- Si se recibe el código de tecla numérica y la señal ENA_CMD está a nivel alto estando en el modo de edición de cada uno de los campos, se activará el bit de la señal de salida LOAD que señala el campo actual en edición y se entregará este código junto con el último número pulsado, o 0 cero en su defecto, a través de la salida DATO_CAMPO. Esto se hará siempre que el valor de dos dígitos introducido sea un minuto o una hora válidos, en función del modo 12h o 24h establecido.
- Se activará la salida CAMBIAR_MODO siempre que se reciba un código de la tecla ‘0xD’ y la señal ENA_CMD esté activa.
- Cuando, estando en el modo programación, no se activen durante más de 7 segundos las señales ENA_CMD o PULSO_LARGO se pasará al modo normal de operación.
- En la salida INFO se informa acerca del estado en el que se encuentra el controlador de acuerdo a la siguiente codificación: "00" normal, "01" editando minutos y "10" editando horas.

Bloque CNT_RELOJ

Entradas: INC_CAMPO, LOAD, DATO_IN, ENA_RELOJ, CAMBIAR_MODO y TIC_1S; ya están descritas en el bloque Control Reloj.

Salidas: SEGUNDOS, MINUTOS, HORAS, MODO y AM_PM; todas ellas descritas en la interfaz del Bloque Global en el apartado 2.

Función: Este bloque implementa la funcionalidad de cuenta inherente a un reloj. Es capaz de llevar la cuenta de las horas, minutos y segundos. La cuenta tiene una habilitación global, ENA_RELOJ, que debe estar a nivel alto para que los contadores se incrementen. El incremento efectivo se producirá cuando se active la señal TIC_1S, que lo hará durante un ciclo de reloj en cada segundo. La señal INC_CAMPO tiene una anchura de dos bits, los cuales provocan, con un nivel alto, el incremento del campo completo de minutos o de horas. La señal LOAD, también de dos bits, valida la carga paralelo del dato que se recibe por la entrada DATO_IN, de 8 bits, en el campo completo de minutos o de horas. La señal CAMBIAR_MODO se utiliza para modificar el formato del reloj entre 12h, con señal de AM_PM adicional para indicar el periodo activo, o 24h.

El diagrama de bloques detallado del bloque CNT_RELOJ se muestra en la figura 6.

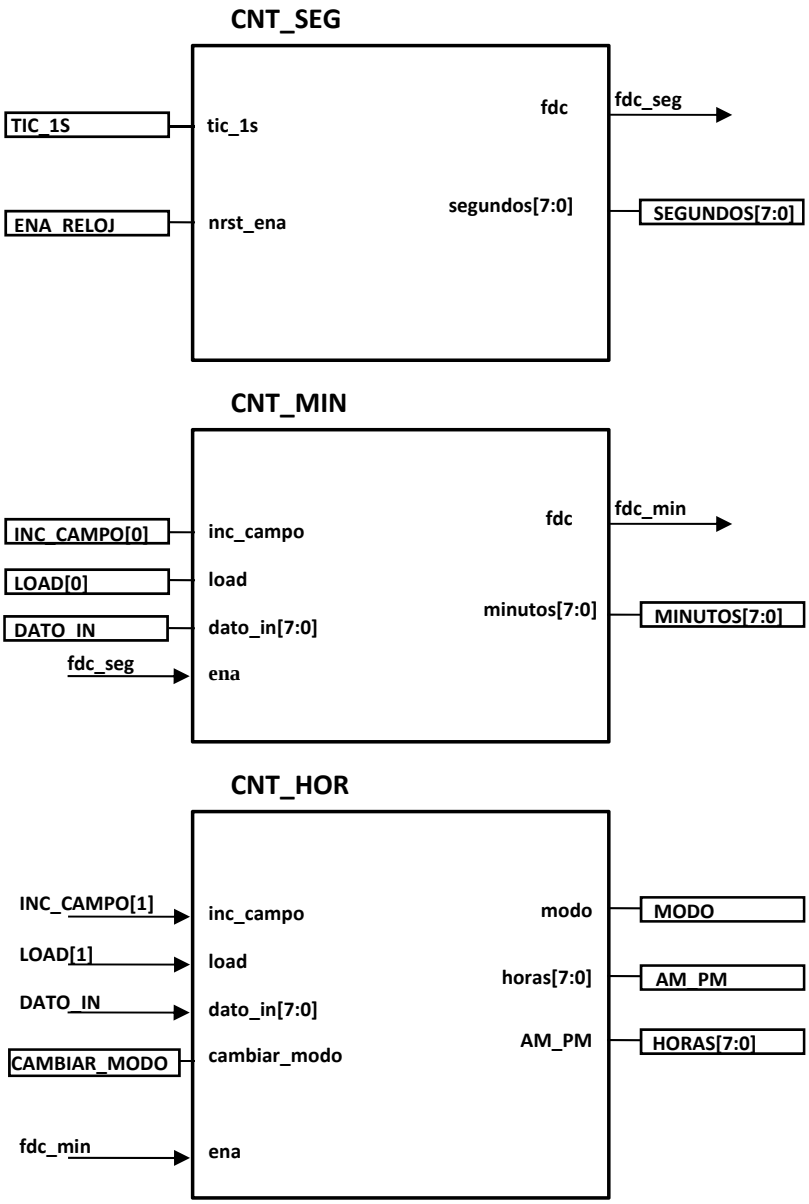


Figura 6

Bloque CNT_SEG

Entradas:

- TIC_1S: entrada de habilitación síncrona, activa a nivel alto, que habilita la cuenta del contador de segundos.
- NRST_ENA: entrada con doble funcionalidad, actúa por una parte como una entrada de habilitación síncrona, activa a nivel alto. Por otra parte, actúa como señal de *reset*, cuando se encuentra a nivel bajo.

Salidas:

- FDC: señal de final de cuenta, que está supeditada a las señales de *enable*.
- SEGUNDOS: señal ya descrita en el apartado 2 en la descripción del bloque RELOJ.

Función: Es un *contador* BCD de dos dígitos de módulo 60. Dispone de señal de fin de cuenta supeditada al *enable*, FDC. Cuenta con una entrada de habilitación, TIC_1S, que habilita la cuenta una vez por segundo. Asimismo, la entrada NRST_ENA, habilita la cuenta cuando está a nivel alto (reloj en marcha) o resetea el contador cuando se encuentra a nivel bajo, de esta forma el contador de segundos comenzará su cuenta a cero tras el proceso de programación de reloj.

Bloque CNT_MIN

Entradas:

- ENA: entrada de habilitación síncrona, activa a nivel alto, que habilita la cuenta del contador de minutos.
- INC_CAMPO: señal de 1 bit que indica con un nivel alto que el campo completo de los minutos debe ser incrementado.
- LOAD: señal de 1 bit que indica con un nivel alto que se cargue el contenido de la entrada "dato_in" en los contadores de unidades y decenas.
- DATO_IN: dato a cargar en los contadores decimales. Los bits más bajos serán el valor a cargar en las unidades, mientras que el valor para las decenas se recibirá por los cuatro bits más altos.

Salidas:

- FDC: señal de final de cuenta, que está supeditada a la señal de ENA.
- MINUTOS: señal ya descrita en el apartado 2 en la descripción del bloque RELOJ.

Función: Es un *contador* BCD de dos dígitos de módulo 60. Dispone de señal de fin de cuenta supeditada al *enable*, FDC. Cuenta con una entrada de habilitación activa a nivel alto, ENA. Dispone de entradas adicionales para el incremento del contador, no supeditadas al *enable*. La señal INC_CAMPO, provoca un incremento del contador BCD de dos dígitos. Es posible cargar en paralelo el valor introducido por la entrada DATO_IN en los contadores de unidades y decenas.

Ejercicio 3

Realice la descripción de la interfaz y función del bloque **CNT_HOR**

6. Diseño lógico y estructural del sistema

Una vez realizado el diseño jerárquico, debe emplearse la documentación generada para describir la estructura de la jerarquía y la interfaz y función de los bloques como especificación para abordar el diseño lógico de cada módulo y la composición del sistema mediante la conexión de éstos. Este trabajo ya ha sido realizado y se encuentra en los ficheros: *reloj.vhd, ctrl_reloj.vhd, cnt_reloj.vhd, cnt_seg.vhd, cnt_min.vhd y cnt_horas.vhd*.

Ejercicio 4

Descargue de *Moodle* y descomprima en su carpeta de trabajo el fichero *BT1_A1_Z1.zip*. Dispondrá de los modelos VHDL de los bloques y los niveles de jerarquía descritos en el apartado 5.
Revise los modelos de la realización del sistema y realice el diagrama de bloques con el que se corresponde cada uno de los sistemas modelados en los ficheros: *ctrl_reloj.vhd, cnt_seg.vhd, cnt_min.vhd y cnt_horas.vhd*.

7. Técnicas de simulación y verificación de circuitos complejos

Los modelos trabajados en el apartado anterior cuentan con algunos errores en el modelado que impiden el cumplimiento de la funcionalidad asignada al bloque. El sistema que viene siendo desarrollado en esta actividad reviste unas complejidad lo suficientemente elevada como para que las técnicas y procedimientos de test que ha venido utilizando hasta ahora no sean suficientes para acometer la simulación y la verificación del funcionamiento de una manera sencilla, organizada y exhaustiva.

Verificar el funcionamiento del reloj supone la introducción de un buen número de comandos y la realización de una serie de acciones secuenciadas que deben llevarse a cabo para efectuar la programación del modo de funcionamiento del reloj. Para tener la seguridad de que el reloj cumple por completo las especificaciones marcadas hay que garantizar que se han probado todas las posibilidades de configuración del reloj y de que este responde correctamente en todas las ocasiones. Por ejemplo, si estamos introduciendo los dígitos de forma individual debemos asegurarnos de que el reloj no queda programado en una hora que no pueda darse.

Así pues, gran parte de las pruebas van a consistir en la simulación de la pulsación de teclas un buen número de veces. Por otra parte, hay que tener en cuenta que la generación de los estímulos necesarios para simular una pulsación es una tarea repetitiva que supone la manipulación de un conjunto de señales de una forma muy parecida. En consecuencia, parece razonable buscar una estrategia que permita optimizar el número de líneas de código del test-bench y evitar la escritura de código excesivamente replicado. El lenguaje VHDL dispone de recursos específicamente concebidos para este propósito: se trata de las sentencias para la creación de subprogramas (funciones y procedimientos).

Por otra parte, analizar el comportamiento del circuito mediante la inspección de los cronogramas como única ayuda es una tarea ingente, tediosa y propensa al error cuando el conjunto de señales a observar es numeroso, como es nuestro caso. En este tipo de tests, resulta de mucha utilidad el empleo de código de autoverificación que monitorice los estímulos proporcionados al sistema y compruebe la corrección de su respuesta. Los bancos de test con autoverificación utilizan sentencias *assert* para informar de los fallos detectados durante el proceso de autoverificación.

En consecuencia, los *test-bench* que se van a utilizar para la simulación y prueba del reloj programable van a descansar significativamente en este tipo de técnicas. En este apartado de la actividad se utilizará un *test-bench* sencillo con autoverificación. Este test le permitirá detectar dos de los errores funcionales que se han cometido en el modelado de los bloques que conforman el reloj programable. Posteriormente, podrá depurar el resto de los errores funcionales utilizando un un *test-bench* más complejo.

Ejercicio 5

Descargue de *Moodle* y descomprima en su carpeta de trabajo el fichero *BT1_A1_Z2.zip*. Dispondrá de los ficheros siguientes: *test_reloj.vhd, pack_test_reloj.vhd, test_estimulos_reloj.vhd y test_monitor_reloj.vhd.* Cree un proyecto e incorpore los ficheros que ha descargado en el apartado 6 de esta actividad. Compile el proyecto completo. No debería producirse ningún error de compilación.

En estos ficheros podrá encontrar el cuerpo principal de un testbench (*test_reloj.vhd*) que emplaza el DUT (*reloj.vhd*), el generador de estímulos (*test_estimulos_reloj.vhd*) y un bloque que contiene el

código de autoverificación del comportamiento funcional del reloj (*test_monitor_reloj.vhd*). El fichero *pack_test_reloj.vhd* contiene un conjunto de funciones y procedimientos útiles para la generación de los estímulos de test y para la elaboración de los monitores que forman el código de autoverificación.

El código que puede encontrar en estos ficheros es extenso, puesto que se utilizan para el test y verificación del modelo completo del reloj. Además observará que algunas partes no están acabadas y deberán ser completadas por el estudiante como parte de su trabajo. Es por ello que no se le pedirá que lo revise de manera completa al principio, sino que irá aproximándose al conocimiento completo del código a lo largo del desarrollo de esta actividad, hasta su finalización. Siga las instrucciones que se ofrecen en los apartados que siguen, mediante las que se irá indicando qué partes del código deberá analizar y completar en cada momento.

En este ejercicio se va a verificar y depurar el funcionamiento correcto del reloj en el modo normal, tanto en el modo 12h, como en el modo 24h. Se comprobará igualmente el paso correcto entre estos dos modos. Para ello, revise de forma general el código que contienen estos ficheros y realice los siguientes apartados en los que se le pedirá un análisis más pormenorizado de algunos extractos de código:

a) Describa qué operaciones se llevan a cabo en la función "*hora_to_natural*" y los procedimientos "*tecleo*", "*cambiar_modo_12_24" y "entrar_modo_prog"* del fichero *pack_test_reloj.vhd.*

b) Escriba el cuerpo del procedimiento "*esperar_hora*" cuya declaración se encuentra en la zona declarativa del paquete y cuya función consiste en esperar hasta que se cumpla la hora exacta (*horas*, *minutos* y *AM_PM*) que se le proporciona como argumentos de entrada (*periodo* y *valor*).

c) Escriba el cuerpo del procedimiento "*fin_prog*" cuya declaración se encuentra en la zona declarativa del paquete y cuya función consiste en simular la introducción del comando de finalización de programación a través del teclado.

d) Analice el código correspondiente a los monitores etiquetados como "monitor 1", "monitor 2", "monitor 3" y "monitor 4" que figuran en el fichero *test_monitor_reloj.vhd* y responda a las cuestiones siguientes.

- Describa qué aspectos del funcionamiento del reloj programable verifica el "monitor 1".

- Describa qué aspectos del funcionamiento del reloj programable verifica el "monitor 2". Sustituya el texto de las sentencias *report* de forma que ofrezcan un mensaje más descriptivo del tipo de error detectado en cada caso.

```
report "Error de tipo 1 detectado por el monitor 2"

report

report "Error de tipo 1 detectado por el monitor 2"

report

report "Error de tipo 2 detectado por el monitor 2"

report
```

- El “monitor 3” pretende verificar el correcto funcionamiento de la señal AM_PM, sin embargo, no se han incluido las sentencias *assert* que se necesitan para ello. Complete con la sentencia apropiada en cada caso.
- Describa qué aspectos del funcionamiento del reloj programable verifica el “monitor 4”.

e) Utilizando los procedimientos que considere necesario, escriba en el fichero *test_estimulos_reloj.vhd* el código que permita testear el funcionamiento correcto del reloj en el modo normal, tanto en el modo 12h como en el modo 24h. Concretamente, los estímulos generados deberán permitir la verificación del incremento correcto de los campos en el recorrido completo del día, la adecuada generación de la señal AM_PM y el cambio correcto de un modo a otro cuando se ordena a través del comando de teclado pertinente. Escriba aquí el plan de test elaborado, de manera previa a su codificación.

f) Lleve a cabo la simulación del circuito analice los resultados obtenidos para detectar los errores que se observan en el funcionamiento del reloj, observe la consola en la que modelsim le notificará la existencia de errores. Realice las modificaciones que crea necesarias para corregir los errores detectados, hasta que la simulación se efectúe sin lanzar por consola ningún mensaje de error.

- Error 1

Describa el error detectado:

Describa la corrección efectuada:

- Error 2

Describa el error detectado:

Describa la corrección efectuada:

- Error 3

  Describa el error detectado:

  Describa la corrección efectuada:

8. Depuración del funcionamiento del diseño lógico

En el apartado anterior trabajó con un *test-bench* simple que le permitió depurar los errores cometidos en el modelado del funcionamiento del reloj en el modo normal. En este apartado se va a

trabajar con el resto de procedimientos y monitores del *test-bench*, a fin de llevar a cabo una simulación más detallada de la operación del reloj. El plan de test del *test-bench* deberá ser ampliado para probar todas las operaciones que pueden ser efectuadas con el reloj en el modo de programación y habrá que proceder a la revisión y completado de código de aquellos monitores que supervisan la correcta operación del modelo. Observará en las nuevas simulaciones, a medida que se vayan incorporando recursos al *test-bench*, que los monitores arrojan en la consola numerosos mensajes de error. El objetivo final es la detección y corrección de todos los errores hasta conseguir que la simulación se realice sin la notificación de ninguna situación anómala. El siguiente conjunto de recomendaciones puede facilitarle el desarrollo de la actividad:

a) Es absolutamente necesario saber qué prueba se está realizando en todo momento y cuál va a ser el procedimiento empleado para detectar los malfuncionamientos del sistema. En consecuencia, antes de proceder con la ampliación del plan de test es preciso analizar el código contenido en los tres ficheros que forman el test hasta su completa compresión, así como la realización de algunos extractos de código que se encuentran ausentes. Concretamente:

   - Analizar el funcionamiento de los procedimientos “time_out” (deja transcurrir tiempo suficiente para que expire el tiempo de timeout de salida del modo de programación) y “programar_hora_inc_corto” (genera estímulos para llevar a cabo la pulsación de las teclas necesarias para programar una hora determinada en el reloj mediante pulsaciones breves del teclado) del paquete pack_test_reloj.
   - Completar el código del procedimiento programar_hora_inc_largo para generar los estímulos correspondientes a la pulsación de las teclas necesarias para programar una hora determinada en el reloj mediante la pulsación larga en la tecla ‘C’
   - Incluir un nuevo procedimiento en el paquete que permita la generación de estímulos para simular la introducción de una hora concreta en base a la pulsación de teclas numéricas. Se recomienda utilizar un procedimiento con la siguiente declaración:

```
-- Programar una hora indicando el valor de cada campo por introduccion numérica
procedure programar_hora_directa
                (signal   ena_cmd:   out std_logic;
                 signal   cmd_tecla: out std_logic_vector(3 downto 0);
                 signal   clk:       in  std_logic;
                 constant valor:     in  std_logic_vector(15 downto 0));
```

   - Analizar los monitores etiquetados como “monitor 7” y “monitor 9”. El primero se encarga de la verificación de salida del modo de programación una vez transcurridos siete segundos de inactividad en el teclado. El segundo monitoriza el incremento del campo en edición, tanto por pulsación breve como larga en la tecla ‘C’.
   - Analizar los monitores etiquetados como “monitor 5” y “monitor 8” y modificar el mensaje lanzado en las sentencias *report* acordes al error concreto detectado en cada caso.
   - Completar el “monitor 6” de forma que haga las funciones de verificación de la salida del modo de programación una vez recibido el comando correspondiente, pulsación breve de la tecla ‘A’.

- Codificar un décimo monitor que llevará las funciones de monitorización de la modificación de la hora del reloj, una vez ésta haya sido introducida mediante la pulsación directa de teclas numéricas.

b) En ocasiones, un determinado problema en el funcionamiento del sistema dificulta el diagnóstico de los demás, es por ello que conviene ir arreglando los errores a medida que se vayan localizando.

c) Para disponer del máximo de información que nos permita detectar con exactitud cuál es la naturaleza del problema con el que nos enfrentamos, es necesario incorporar a la ventana de formas de onda no solamente las señales de la interfaz global del bloque, sino también las señales de los sub-bloques que forman toda la jerarquía del diseño.

d) El mensaje que puede leerse en la consola, es simplemente la evidencia de que algo anormal está sucediendo y puede darnos pistas de por dónde debemos empezar a revisar los modelos. Sin embargo, la información más útil a la hora de averiguar qué está mal modelado y cómo debe ser corregido nos la va a aportar la observación concienzuda del cronograma resultado de la simulación.

SOLUCIONES DE LOS EJERCICIOS 1, 2 Y 3

Ejercicio 1

Incremento de las horas cuando proceda.
Incrementar en la unidad el valor de las horas.
Cargar un valor concreto en los dígitos de unidades y decenas
Ofrecer una salida conforme al modo 12h o 24h
Indicar el periodo horario AM o PM en el modo 12h

Ejercicio 2

*"Cuando el usuario pulsa la tecla 'A', el sistema externo al bloque que estamos considerando la procesa y genera un pulso alto en la señal **ena_cmd** de un ciclo de reloj de duración, al tiempo que mantiene el código hexadecimal 0xA en la señal **cmd_tecla**. En esta circunstancia el autómata pasará al estado normal, lo cual será advertido mediante el cambio en la señal **info**. Por otra parte, deberá garantizarse que la señal **ena_reloj** se pone a nivel alto para permitir el funcionamiento del reloj con normalidad. El bloque CNT_RELOJ al encontrar su entrada **ena_reloj** a nivel alto incrementará el tiempo medido con normalidad"*

Ejercicio 3

Bloque CNT_HOR

Entradas:

- ENA: entrada de habilitación síncrona, activa a nivel alto, que habilita la cuenta del contador de horas.
- INC_CAMPO: señal de 1 bit que indica con un nivel alto que el campo completo de las horas debe ser incrementado.
- LOAD: señal de 1 bit que indica que debe ser cargado el contenido de la entrada "dato_in" en los contadores.
- DATO_IN: dato a cargar en los dos dígitos correspondientes a las horas. Los bits más bajos serán el valor a cargar en las unidades, mientras que el valor para las decenas se recibirá por los cuatro bits más altos.
- CAMBIAR_MODO: señal de un bit, activa a nivel alto, que provoca el cambio entre el modo 12h y el modo 24h, y viceversa.

Salidas:

- MODO: salida de un bit que indica el modo activo (0: 12h; 1; 24h)
- AM_PM: salida de un bit que indica el periodo horario activo (0: AM; 1: PM)
- HORAS: señal ya descrita en el apartado 2en la descripción del bloque RELOJ.

Función: Es un *contador* BCD de dos dígitos de módulo 24. Cuenta con una entrada de habilitación activa a nivel alto, ENA. Dispone de entradas adicionales para el incremento del contador, no supeditadas al *enable*. La señal INC_CAMPO, provoca un incremento del contador BCD de dos dígitos. Es posible cargar en paralelo el valor introducido por la entrada DATO_IN en los contadores de unidades y decenas de hora. Dispone de dos modos de funcionamiento, 12h y 24h, transitará de uno a otro cuando se active la entrada CAMBIAR_MODO. El modo activo se reflejará en todo momento en su salida MODO. En el modo 12h, el contador será de módulo 12 y cuando se alcance el último estado de cuenta se conmutará entre los periodos AM y PM de manera automática. El periodo horario activo será reflejado en todo momento en la salida AM_PM.