Prototipo de microcontrolador de aplicación específica

1. Objetivos

En su primera fase de prototipado, un microcontrolador puede ser implementado en VHDL a fin de comprobar su funcionalidad. En la práctica que proponemos en este enunciado se lleva a cabo el desarrollo de un microcontrolador de propósito específico. Este microcontrolador poseerá una arquitectura muy simplificada y un juego de instrucciones reducido, de forma que, mediante un simple programa de código máquina, sea posible controlar algunos elementos (interruptores, actuadores de nivel, termostato, etc.). Como consecuencia de la radical simplificación del procesador se ha suprimido el control de interrupciones, por lo que la llegada de nuevos datos por medio de los periféricos debe comprobarse dentro del programa ensamblado.

El prototipo que nos ocupa debe ser capaz de recibir órdenes simples a través de una interfaz serie RS232, así como enviar información sencilla a través de esta línea. **El control físico de la línea es el realizado en la Práctica II**. Los datos recibidos se almacenarán en una memoria RAM, para lo cual se debe desarrollar un sistema de control de acceso directo a memoria que pida los buses al procesador cuando una transferencia sea necesaria. Para la transmisión se habilitará un registro especial cuyo control será compartido por el control de acceso a memoria y el núcleo del procesador. En esta fase preliminar, la interfaz serie estará regida en su otro extremo por un PC con salida serie que ejecuta un programa de terminal.

La arquitectura interna del procesador es Harvard, es decir, dispone de memoria de datos y de programa con buses separados. El control dispone de una memoria ROM de programa, una RAM de datos con una estructura específica, una unidad ALU y el ya mencionado controlador RS232.

El propósito del microcontrolador objeto de la práctica es decodificar los comandos llegados por el puerto serie y actuar en consecuencia, bien sea enviando información por el puerto serie, bien actuando sobre una serie de interruptores y otros periféricos. Los periféricos que serán gestionados por el procesador son:

- 8 interruptores ON/OFF
- 10 actuadores de nivel
- Un termostato

2. Contenidos

En esta práctica se pretende construir los bloques que constituyen un microcontrolador de aplicación específica. En este documento se presentan las especificaciones del sistema y se propone una arquitectura determinada. Además, se indica una metodología para completar la descripción del sistema. En cualquier caso, **el alumno es libre de modificar todos los ficheros que se entregan al principio de la práctica**, según vaya realizando los distintos pasos, o para implementar las mejoras que considere oportunas en el sistema.

3. Resumen del estándar RS232

El estándar de comunicación asíncrona por línea serie RS232 ha sido uno de los más extendidos en el desarrollo de periféricos. Este estándar permite la comunicación bidireccional por dos líneas separadas, e incluye líneas opcionales de control de flujo de datos y negociación de transferencias.

La versión más simple del estándar especifica que la línea de transmisión de datos debe permanecer a nivel alto en reposo, iniciándose la transmisión con un bit que siempre será un nivel bajo. La forma más simple de transmisión envía 8 bits de datos y uno de parada (nivel alto), sin añadir bits de paridad, tal y como se indica en la Figura 1. La velocidad de transmisión puede configurarse desde 9.600 bps hasta 115.200 bps.

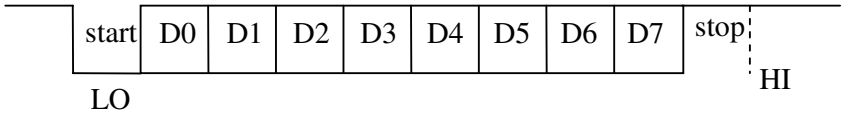

Figura 1. Protocolo de envío RS232

En el sistema propuesto en la práctica se hace uso de un controlador del nivel físico que emplea este modo de transmisión simple a una velocidad de 115.200 bps. Este sistema, al igual que el resto del procesador, será alimentado con una señal de reloj de 20 MHz, una frecuencia suficientemente alta para la funcionalidad del procesador.

4. Descripción de los comandos del terminal

Tal y como se ha explicado, la principal tarea del sistema microcontrolador es decodificar una serie de comandos recibidos a través de una línea serie. De acuerdo con los periféricos que se van a controlar y sus posibles valores, los comandos que puede recibir el procesador por la línea serie son los indicados en la Tabla 1.

Comando	Parámetro 1	Parámetro 2	Descripción
I	0..7	0, 1	Selecciona un interruptor de los 8 existentes y lo enciende (1) o lo apaga (0)
A	0..9	0..9	Selecciona uno de los 10 actuadores y le asigna un valor de apertura de 0 a 9
T	1, 2	0..9	Carga el valor del termostato a una temperatura entre 10 y 29 grados
S	I, A, T	0..9	Solicita información al procesador sobre el estado de alguno de los periféricos

Tabla 1. Comandos del terminal

Nótese que todos estos comandos están escritos en mayúscula, por lo que el procesador sólo responderá a los códigos ASCII de los caracteres en mayúscula. Asimismo es de notar que los comandos no se validan con un retorno de carro, lo que se debe a que este procesador no debería ser accedido directamente por un terminal, sino a través de una consola de mandos más elaborada. El sistema que se trata de diseñar en esta práctica es un prototipo, por lo que no nos preocupa acceder con una herramienta más pobre a fin de comprobar su funcionalidad.

El último de los comandos forzará al procesador a enviar a través de la línea serie dos bytes con información acerca del estado del periférico indicado. Este comando incluye dos parámetros: en el primero se indica el tipo de periférico del que se solicita información; en el segundo se indica el

número identificador del periférico. Nótese que en este caso el parámetro 2 sólo tiene sentido para los interruptores y los actuadores, pudiendo tomar cualquier valor en el resto de casos.

Para la decodificación de estos comandos **se proporcionará a los alumnos un programa escrito en el ensamblador del procesador**, y que constituye una versión simple del programa que deberá ejecutar el sistema. Si los alumnos desean modificar y mejorar este programa, se pondrá a su disposición un software que compile el código y lo convierta en una ROM para VHDL. Nótese sin embargo que la modificación del programa es, para los objetivos de la práctica, un aspecto **opcional**.

5. Descripción de la arquitectura y de la interfaz con el exterior

El sistema microprocesador objeto de la práctica tiene una arquitectura Harvard, es decir, los buses de datos e instrucciones están separados. **El bus de instrucciones está disponible únicamente para el decodificador** de las mismas, mientras que **del bus de datos cuelgan, además del control, la memoria RAM, la ALU y el sistema de DMA**. Según lo descrito hasta el momento, la arquitectura puede resumirse en la Figura 2.

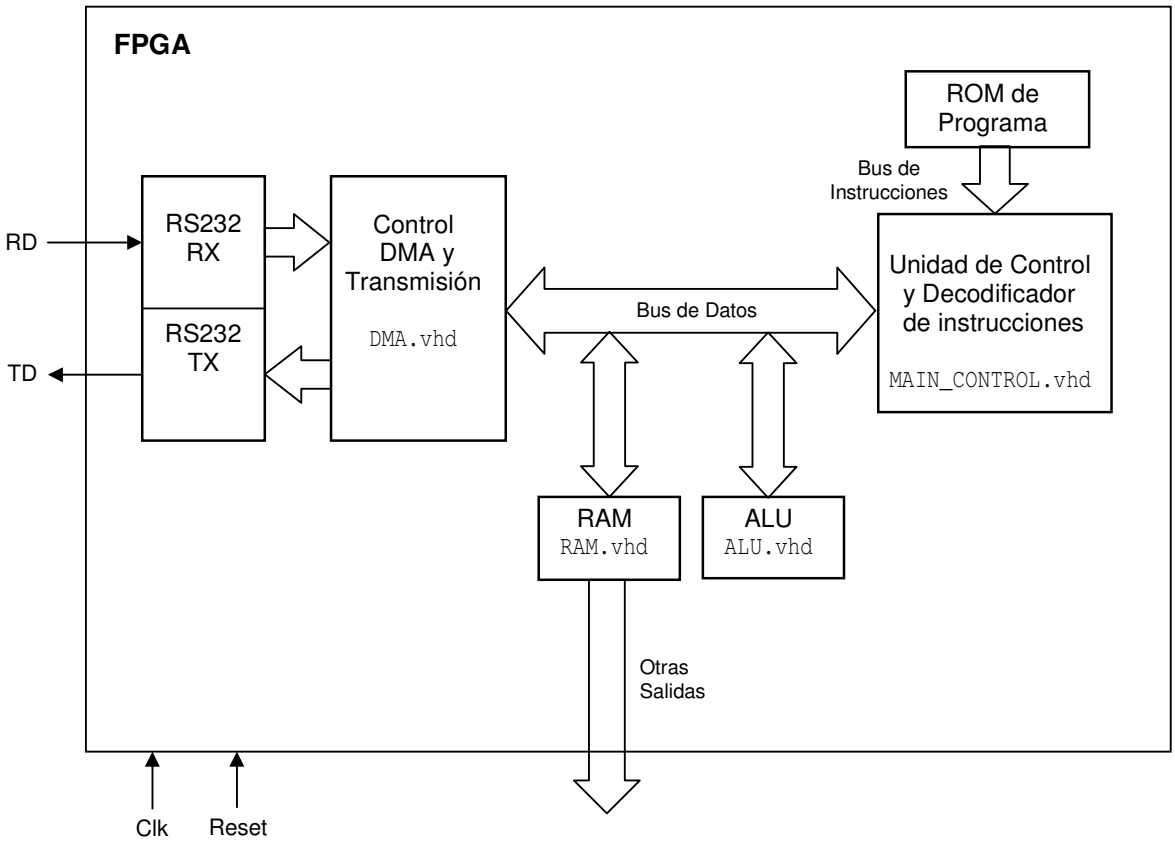

Figura 2. Diagrama de bloques de alto nivel

Como puede verse en la figura, el procesador dispone de apenas un pequeño conjunto de líneas de entrada y salida. Aparte de las líneas de reloj y reset, y de las de transmisión/recepción RS232, el sistema dispondrá de un grupo de líneas desde la RAM que aprovechen el hardware montado en el laboratorio para mostrar información de interés. Esta información estará relacionada con el estado de cada uno de los periféricos que se controlan.

El diagrama se representa a muy alto nivel, y no muestra más que las conexiones lógicas entre los bloques del sistema. La descripción de las entradas, salidas, y funcionalidad de cada bloque se indicará a continuación, cuando se describa cada uno de ellos.

6. Descripción funcional de los bloques del sistema

6.1. Interfaz física RS232

La implementación de la interfaz física de transmisión y recepción por línea RS232, realizada en la Práctica II, se caracteriza por:

- Recepción y transmisión de datos a 115.200 bps.
- Formato de trama de 8 bits de datos, sin paridad y con un bit de parada.
- Capacidad de almacenamiento de datos recibidos de hasta 16 bytes mediante una memoria FIFO interna.
- Líneas de estado de la memoria interna de recepción: indicadores de memoria llena y memoria vacía.
- Entrega de datos recibidos por una interfaz paralelo y síncrona mediante una línea de Data_read.
- Captura de datos mediante una interfaz asíncrona basada en una línea de validación de datos (Valid_D) y una de reconocimiento de carga (ACK_in).
- Línea de estado del transmisor: reposo o enviando.

Las líneas de entrada y salida de la interfaz se indican, junto con su descripción, en la Tabla 2.

LÍNEA	SENTIDO	FUNCIÓN
Reset	Entrada	Reset asíncrono y activo a nivel bajo.
Clk	Entrada	Reloj del sistema.
Data_in [7..0]	Entrada	Entrada de los datos que el sistema cliente desea enviar. Bit más significativo en la posición 7.
Valid_D	Entrada	Validación del dato de entrada por parte del sistema cliente. Activa a nivel bajo.
ACK_in	Salida	Reconocimiento del dato para transmitir. Activa a nivel bajo, se activa al guardar un nuevo dato válido y se desactiva al desactivarse Valid_D.
TX_RDY	Salida	Transmisor está disponible. Activa a nivel alto.
TD	Salida	Línea de transmisión RS232.
RD	Entrada	Línea de recepción RS232.
Data_out [7..0]	Salida	Datos de salida de la FIFO interna del subsistema receptor. Bit más significativo en la posición 7.
Data_read	Entrada	Petición del sistema cliente de lectura de un nuevo dato de los recibidos y almacenados. Activa a nivel alto y síncrona con el reloj.
Full	Salida	Indica que la memoria interna del subsistema receptor está llena. Activa a nivel alto.
Empty	Salida	Indica que la memoria interna del subsistema receptor está vacía. Activa a nivel alto.

Tabla 2. Líneas de entrada y salida de la interfaz RS232

Como puede inferirse de la tabla anterior, el protocolo de recepción de datos es simple. El sistema cliente debe muestrear la línea EMPTY y, cuando esté a nivel bajo, activar la línea Data_read durante un ciclo de reloj para recibir los datos por las líneas Data_out.

Por otra parte, la carga de datos para su envío debe tener en cuenta que nunca se debe intentar cargar un nuevo dato mientras el transmisor esté ocupado, ya que de ese modo variaríamos los datos que se están enviando. Afortunadamente esto queda solucionado mediante el protocolo de carga de datos

asíncrono de que se dispone. La interfaz ofrecida nunca carga el nuevo dato mientras el transmisor no esté libre. Por ello, para cargar un dato, se debe colocar éste en el bus Data_in, poner a nivel bajo la línea Valid_D y esperar que se ponga a nivel bajo la línea ACK_in. La línea TX_RDY se ofrece a la salida como información redundante por si fuera necesario hacer uso de ella para ejecutar otra tarea. La temporización de la entrega de datos del DMA al transmisor RS232 se muestra en la Figura 3.

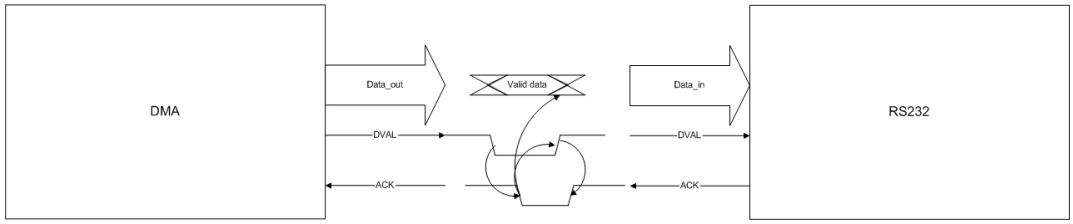

Figura 3. Protocolo de entrega de datos al transmisor RS232

6.2. Arquitectura de la memoria de datos

En este primer prototipo del microcontrolador utilizaremos una entidad RAM que funcionará a modo de "cajón de sastre". En esta fase preliminar, dicha entidad no sólo implementará la memoria RAM de propósito general, sino también una serie de posiciones de memoria (registros de E/S) que simulan los diferentes periféricos que el sistema completo debe controlar, así como las posiciones de memoria empleadas por el controlador de acceso directo a memoria. No contentos con eso, vamos a añadir una serie de líneas de salida que aprovechan los recursos disponibles en el hardware del laboratorio, a fin de visualizar el estado de alguno de esos periféricos.

Hechas estas consideraciones, el mapa de la memoria de datos del procesador aparece en la Tabla 3.

Dirección	Alias	Función
0x00	DMA_RX_Buffer (MSB)	Byte más significativo de la reserva para el controlador DMA (recepción)
0x01	DMA_RX_Buffer	Byte intermedio de la reserva para el controlador DMA (recepción)
0x02	DMA_RX_Buffer (LSB)	Byte menos significativo de la reserva para el controlador DMA (recepción)
0x03	NEW_INST	Flag que indica la llegada de un nuevo comando por la línea serie
0x04	DMA_TX_Buffer (MSB)	Byte más significativo de la reserva para el controlador DMA (transmisión)
0x05	DMA_TX_Buffer (LSB)	Byte menos significativo de la reserva para el controlador DMA (transmisión)
0x06 … 0x0F	Reservado	Para posterior ampliación
0x10 … 0x17	SWITCH(0..7)	Zona de control de interruptores
0x18 … 0x1F	Reservado	Para posterior ampliación
0x20 … 0x29	LEVER(0..9)	Zona de control de actuadores
0x2A … 0x30	Reservado	Para posterior ampliación

0x31	T_STAT	Temperatura fijada en el termostato
0x32 … 0x3F	Reservado	Para posterior ampliación
0x40 … 0xFF	GP_RAM	Memoria de propósito general

Tabla 3. Mapa de memoria de datos

La entidad RAM que implementa el mapa de memoria está conectada al bus de datos general del procesador, **por lo que debe disponer de salidas triestado**. Las líneas de acceso al componente se especifican en la Tabla 4.

Línea	**Sentido**	**Descripción**
Reset	Entrada	Reset asíncrono a nivel bajo para los registros de E/S
Clk	Entrada	Reloj principal del sistema (20MHz)
Databus[7..0]	Bidireccional	Bus de datos del sistema
Address[7..0]	Entrada	Direcciones del bus de datos
Write_en	Entrada	Habilitación de escritura
OE	Entrada	Habilitación de lectura
Switches[7..0]	Salida	Estado de los interruptores
Temp_L[6..0]	Salida	Dígito más bajo del valor de la temperatura del termostato (formato 7 segmentos)
Temp_H[6..0]	Salida	Dígito más alto del valor de la temperatura del termostato (formato 7 segmentos)

Tabla 4. Líneas de entrada y salida de la RAM

La escritura de la RAM se realiza de forma síncrona, y es habilitada mediante la señal Write_en. La señal de Reset existe como forma de inicialización de los registros de carácter específico de la RAM (direcciones 0x00 a la 0x3F). El Reset debe inicializar dichos registros a un valor conocido, aunque no necesariamente nulo (carecería de sentido en el caso del termostato).

Las líneas más especiales de la RAM son las de salida al exterior del sistema. Switches[7..0] debe recoger el estado de los interruptores mapeados en memoria para su presentación en la barra de LEDs de que dispone el hardware del laboratorio. Del mismo modo, las líneas Temp_X[7..0] deben presentar la conversión del valor guardado en el termostato de forma que sea inteligible en dos visualizadores de 7 segmentos.

8. Consideraciones adicionales para la implementación

8.1. Memoria RAM

La memoria RAM del procesador puede dividirse en dos bloques de cara a su implementación. En primer lugar la batería de posiciones de memoria que poseen algún significado específico (las posiciones 0x00 a 0x3F), y por otro lado la memoria de propósito general (posiciones 0x40 a 0xFF). Atendiendo a su descripción funcional, la única diferencia apreciable entre una zona de memoria y la otra es la existencia de una señal de Reset para la primera. Esta señal se provee como ayuda para agilizar la inicialización del sistema y reducir la memoria de programa, pero podría eliminarse y realizar esta función con software. Para la implementación de este bloque se ofrece como ejemplo el archivo `RAM.vhd`, que constituye una memoria de propósito general de 16 bytes.

La tarea del alumno con este bloque es crear una entidad que contenga una memoria RAM de propósito general como la ofrecida, pero del tamaño adecuado, y una segunda memoria que contenga la señal de Reset. La memoria de carácter general deberá construirse en una entidad aparte (bastará con

modificar adecuadamente el fichero `RAM.vhd`), siendo luego instanciada en el bloque principal de la memoria.

La ventaja de trabajar con la memoria RAM genérica por separado es que la herramienta de síntesis la reconocerá como tal, y será capaz de optimizarla haciendo un mejor uso del hardware. Esto no ocurre sin embargo con la zona de memoria con señal de Reset, para la que el sintetizador deberá hacer uso de flip-flops discretos hasta completarla. Evidentemente, el alumno debe ser capaz de decodificar la dirección especificada para acceder a uno u otro banco de memoria.

La RAM dispone de líneas de salida destinadas a mostrar el estado de alguno de los registros de propósito específico. Las líneas SWITCHES muestran el estado de los interruptores, para lo que basta que cada línea esté directamente asociada con el bit menos significativo de cada registro de estado de interruptor. Las líneas TEMP_L y TEMP_H deben convertir el '*nibble*' bajo y alto del registro T_STAT (dirección 0x31) a un formato legible en un display de 7 segmentos. Esto puede conseguirse de forma sencilla construyendo una ROM con una instrucción del tipo with-select (aunque existen otras maneras más o menos elegantes de conseguir el mismo resultado). Para la construcción de estas funciones de conversión, téngase en cuenta que las líneas de los visualizadores están 'mapeados' con respecto a las líneas de salida según lo indicado en Figura 4.

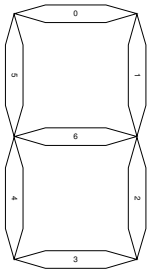

Figura 4. Líneas de los displays de 7 segmentos

Una última consideración a tener en cuenta es el hecho de que **el bus de datos debe permanecer en estado de alta impedancia mientras que la señal OE esté a nivel bajo** (véase la especificación de las señales en la Tabla 4).

Como apoyo para la codificación de la RAM se han definido algunos tipos de datos en el paquete `PIC_pkg` (contenido en el archivo `PIC_pkg.vhd`). Este paquete contiene, aparte de los ya mencionados tipos, otras funciones y tipos destinados a facilitar el desarrollo de la práctica.