|  |
| --- |
| universidad politécnica de madrid |
| Memoria proyecto LCSE |
| M.I.S.E. |
|  |
| **Rodolfo B. Oporto Quisbert** |
| **1/20/2016** |

|  |
| --- |
|  |

Contenido

[Etapa 1: RS-232, RAM y DMA 2](#_Toc441091662)

[RS\_232 2](#_Toc441091663)

[RAM 2](#_Toc441091664)

[Memoria RAM de periféricos 2](#_Toc441091665)

[Memoria RAM de propósito general 3](#_Toc441091666)

[Controlador DMA 3](#_Toc441091667)

[Simulación Post Route 4](#_Toc441091668)

[Etapa 2: ALU, ROM y Controlador principal 5](#_Toc441091669)

[ALU 5](#_Toc441091670)

[CONTROLADOR 5](#_Toc441091671)

[Bloque de control 6](#_Toc441091672)

[Etapa 3: Sistema completo 7](#_Toc441091673)

# 

# Etapa 1: RS-232, RAM y DMA

## RS\_232

Para el controlador RS-232 (probado anteriormente en placa), realizamos las siguientes modificaciones :

* Se eliminaron las salidas registradas para evitar retardos: Detectamos que había un problema de temporización a la hora de registrar las salidas del módulo RS-232, dado que no tenemos un control de flujo implementado entre módulos y esto suponía retardos a la hora de leer y escribir por el RS-232.

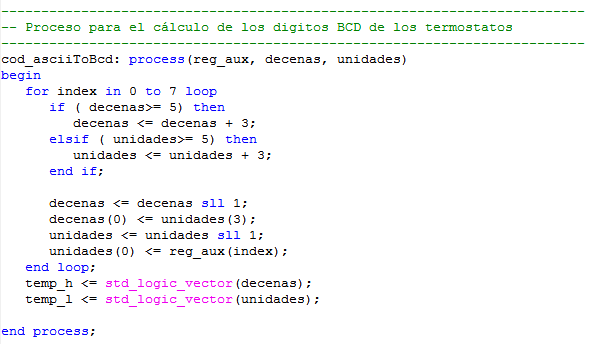
## RAM

Para la implementación de la memoria RAM, hemos dividido este módulo en dos sub-módulos:

* Memoria RAM de periféricos, con Reset asíncrono activo a nivel bajo y que además mapea todos los periféricos, en 64 bytes.
* Memoria RAM de propósito general de 192 bytes.

### Memoria RAM de periféricos

En esta entidad aparte de toda la lógica para la lectura y escritura de la RAM, tenemos un proceso que se encarga de generar el código BDC para los displays de 7 segmentos.



Este proceso es sensible a reg\_aux, que es un registro auxiliar interno que guarda el contenido de la dirección 49 en la que se encuentra el valor del termostato y que se registra cada ciclo de reloj, dentro del proceso que controla la lectura y escritura.

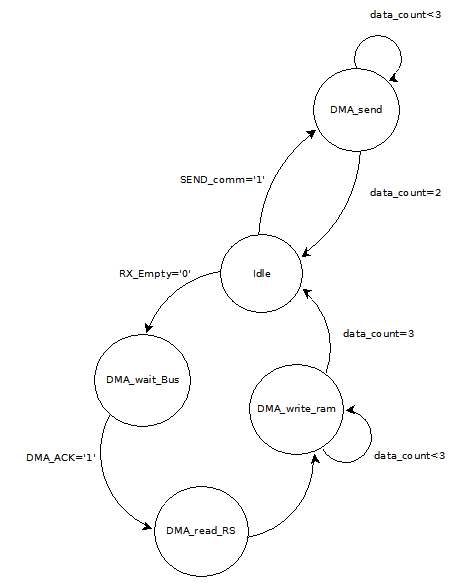


### Memoria RAM de propósito general

Este bloque de memoria es similar al anterior, con la salvedad que no posee un reset y no mapea en las salidas el valor de ninguna posición de memoria, como ocurre con la RAM de periféricos.

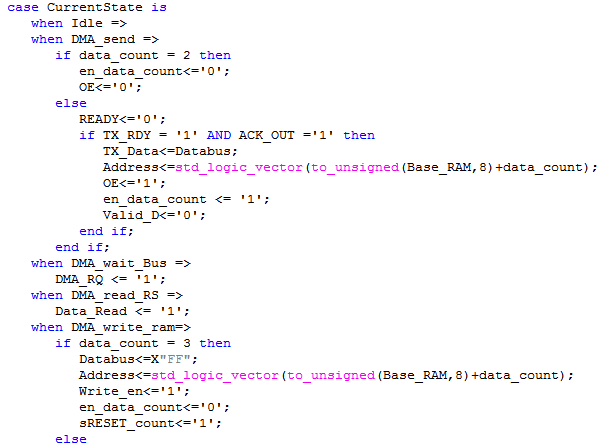
## Controlador DMA

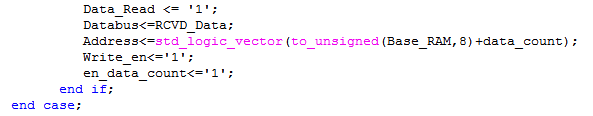
Para el diseño del controlador DMA, implementamos la siguiente máquina de estados:



También se implementa todas las señales necesarias para controlar el RS-232 y acceder a la memoria RAM

Está pensada como una máquina de Mealy, dado que depende del estado actual y de otras señales (externas e internas) como el contador de datos, para la generación de las direcciones que de acceso a la RAM.

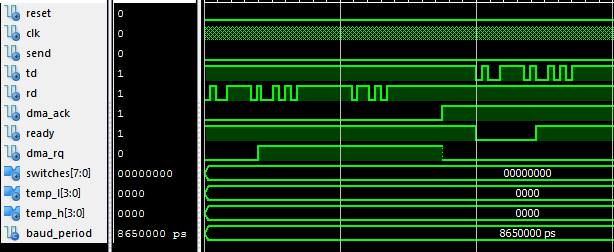




## Simulación Post Route

El fichero tb\_RS232\_DMA\_RAM.vhd implemente el testbench implementado para validar esta etapa.

Resulta tan sencillo como escribir tres datos en la memoria en las posiciones 0, 1, 2, 3 (FF como dice el enunciado) y leer las posiciones de memoria (0 y 1 en este test, dado que la dirección de memoria de respuesta 4 y 5 no contienen nada a priori).



Podemos apreciar cómo se reciben los valores 10011110(@0) y 01010111(@1) y 10110101(@2) , se solicita el bus de datos con DMA\_RQ y cuando el controlador lo concede con DMA\_ACK, se procede a leer las posiciones de memoria @0 y @1 para enviarlas por TD.

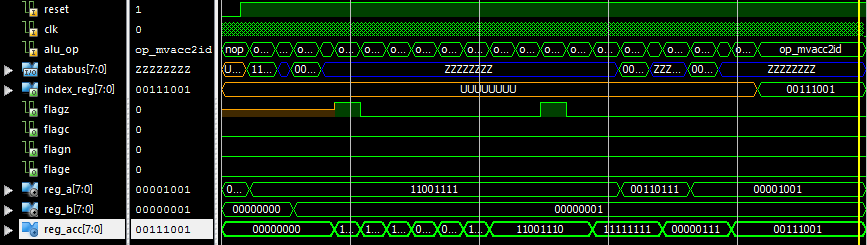
Todos los ficheros necesarios para reproducir esta simulación están en la carpeta ETAPA1 DMA, RAM y RS232

# Etapa 2: ALU, ROM y Controlador principal

## ALU

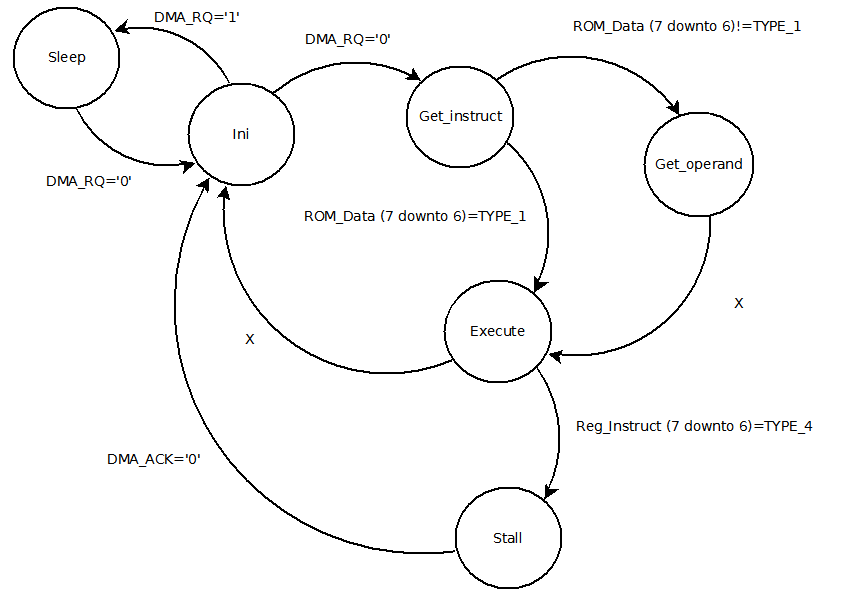
Para la implementación de la ALU nos decantamos por un circuito puramente combinacional, esto nos obligará a controlar muy bien los accesos y dejar la ALU en estado de NO OPERACIÓN cuando no se esté utilizando.

Para validar la funcionalidad de la ALU, se escribió el fichero tb\_ALU.vhd que prueba todas las operaciones implementadas.



## CONTROLADOR

Para el controlador, hemos implementado el siguiente autómata:



Para esta parte, fue necesaria la implementación de un decodificador de instrucciones que activa las micro-órdenes que deben ejecutarse, dentro del estado Execute.

## Bloque de control

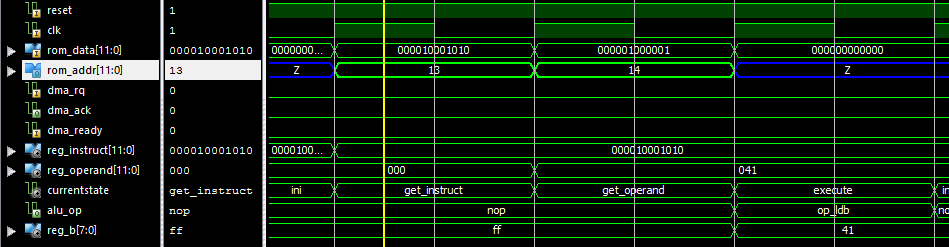
Este bloque no está propuesto en el enunciado, pero engloba la ALU, la ROM y el controlador. Se implementó como paso previo a la integración total y para tener un diseño más modular.

El testbench tb\_bloque\_control.vhd realiza la ejecución de las primeras instrucciones de la ROM. Para comprobar que se guardan correctamente las instrucciones, operandos y se generan las micro-ordenes.

Por ejemplo, en la instrucción 13, tenemos:



Esta instrucción carga el literal X41 en el registro B de la ALU.



Como podemos apreciar en la captura, se realiza correctamente.

# Etapa 3: Sistema completo

A la hora de integrar existen problemas debidos a la lectura de la RAM, que no hemos sido capaces de solventar hasta la entrega.

* Una vez realizada la lectura de la RAM, esta no se muestra inmediatamente y se muestra el próximo ciclo, cargando en el registro de envío por el RS-232, un valor erróneo.

De todas maneras, se adjuntan los ficheros que componen todo el bloque.

Esperamos entregar una versión funcional para presentarla durante la entrevista de la entrega.