MEMORIA PROYECTO 2

BUS COMPARTIDO Y DMA

DANIEL RUEDA MACÍAS

559207

Universidad Zaragoza

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

Contenido

RESUMEN

En este proyecto se ha realizado la programación, a través de una máquina de estados Mealy, de la unidad de control de un DMA, el cual es el controlador de un periférico que comparte el bus de memoria con el MIPS del proyecto anterior. Se han tenido que gestionar los accesos al bus para que no haya conflictos y parar el procesador en caso de que una transferencia en modo ráfaga esté en proceso. Por último, se ha realizado el apartado optativo, que consiste en añadir a la máquina de estados de la UC del DMA un modo robo de ciclo, para así poder aumentar la eficiencia del procesador, ya que, si el MIPS se para por cada transferencia, el tiempo de ejecución del programa que se esté ejecutando se disparará.

INTRODUCCIÓN

Se tiene el MIPS de la práctica anterior, con sus instrucciones LW y SW pre-incremento, su unidad de detención ante riesgos de datos y su predictor de saltos. Se ha sustituido el bus de memoria de datos de dicho procesador por un bus en el que están conectados la MD con su controlador y el periférico IO con su respectivo controlador (DMA).

Este bus es un bus semi-síncrono que soporta ráfagas de tamaño variable. Respecto al arbitraje, el MIPS siempre actúa como master, el DMA puede actuar como master o como slave y la MD siempre como slave. El bus incluye una línea MIPS_REQ que activa el controlador del MIPS cuando desea utilizar el bus. Si el bus no está realizando una ráfaga y MIPS_REQ está activo la CPU tendrá el uso del bus. En caso contrario el DMA podrá utilizar el bus. Si el DMA comienza una ráfaga, tendrá el uso del bus hasta que la ráfaga termine.

Por último pero no menos importante, DMA e IO están comunicados por un bus asíncrono. Esto quiere decir que para que el DMA realice lecturas o escrituras en IO, estos dos se tienen que poner de acuerdo con sus respectivas señales de sincronización: *DMA_sync e IO_sync*.

OBJETIVOS

Al final se quiere llegar a que el DMA realice transferencias entre MD e IO, leyendo de MD y escribiendo en IO o viceversa. En modo ráfaga o en modo robo de ciclo. El MIPS da la orden de transferencia al DMA y este se encarga de realizarla mientras el MIPS sigue por su cuenta realizando otras operaciones.

Para ello, es de vital importancia realizar una correcta máquina de estados Mealy que cumpla todos estos requisitos.

METODOLOGÍA

Este apartado de metodología se va a dividir en 4 partes:

1. Cambios realizados en el DMA.
2. Parada del MIPS.
3. Máquina de estados de la unidad de control del DMA.
4. Pruebas realizadas.

Cambios realizados en el DMA

El DMA que se proporciona inicialmente consta de un registro de control que guarda la orden de la transferencia, un contador en el que se lleva la cuenta del número de palabras transferidas, un registro de datos en el cual se guarda la palabra que se está transfiriendo para mandarla luego a IO o a MD según lo que toque y la unidad de control.

Quitando la UC, para realizar transferencias en modo ráfaga no hay que modificar nada del DMA, pero para poder hacer transferencias en modo robo de ciclo se han realizado las siguientes modificaciones:

- Se ha creado la señal *robo* que indica a la UC que la transferencia se realizará en modo robo de ciclo. El valor de dicha señal se obtiene del bit 26 del registro de control. Si la señal vale 1 la transferencia se realizará en modo robo de ciclo, en caso contrario en modo ráfaga. Por último, se ha conectado como señal de entrada a la UC.
- En las transferencias en modo ráfaga, el controlador de la MD va actualizando las direcciones de memoria en cada ciclo para cuando se desee, leer o escribir palabras en modo ráfaga. Pero para poder realizar una transferencia en modo robo de ciclo, el DMA tiene que encargarse de llevar la cuenta de las direcciones para poder ir pidiéndolas cuando el bus que le comunica con MD esté libre. Por tanto, se ha cambiado la sentencia concurrente que ponía en el bus la dirección inicial de la primera palabra por la dirección inicial más el número de palabras en caso de que la señal *robo* esté activa, en caso contrario sigue introduciéndose la dirección de la palabra inicial.

Parada del MIPS

Para que el MIPS pare cuando se hacen las transferencias en modo ráfaga, se ha llevado el código de operación hasta la etapa MEM y se han definido los siguientes riesgos:

- Que se esté utilizando el bus y que la instrucción que haya en MEM sea un LW.
- Que se esté utilizando el bus y que la instrucción que haya en MEM sea un LW_pre.
- Que se esté utilizando el bus y que la instrucción que haya en MEM sea un SW.
- Que se esté utilizando el bus y que la instrucción que haya en MEM sea un SW_pre.

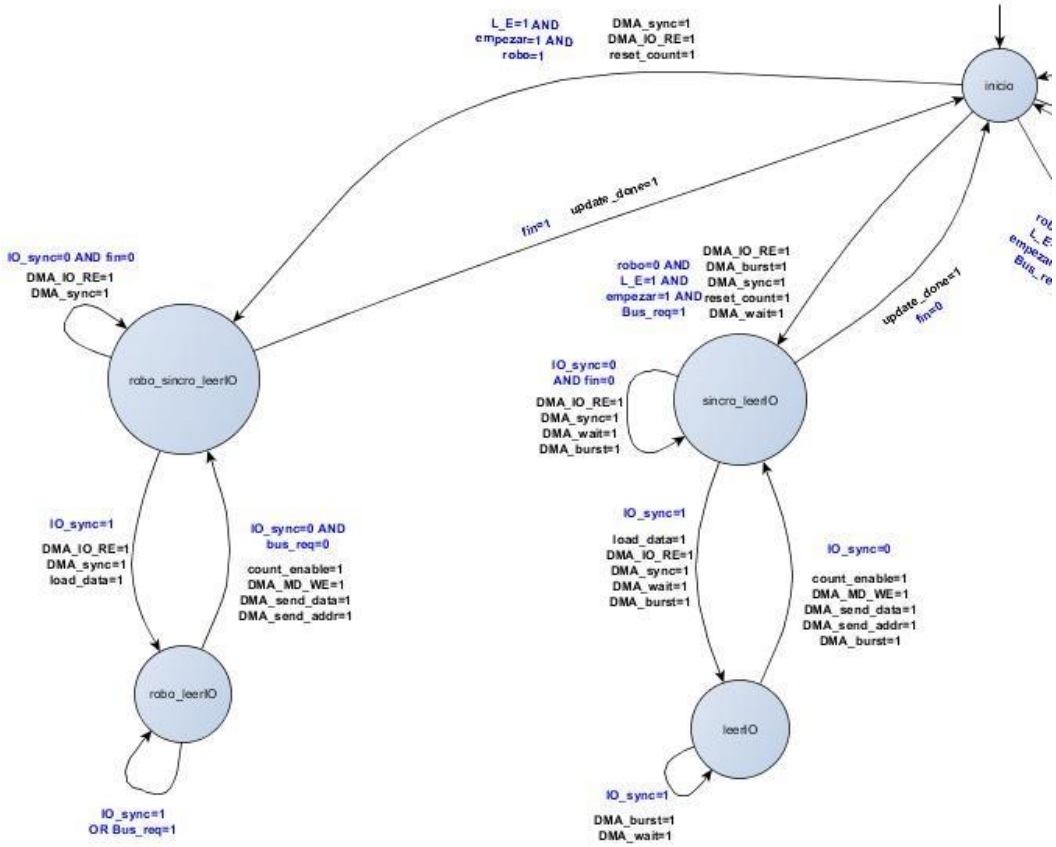
Acto seguido se ha condicionado la señal de carga de los bancos ID_EX y EX_MEM si se da alguno de estos riesgos y la del PC y el banco IF_ID si se da alguno de los riesgos de datos o riesgos de bus.

Por último, se han puesto las señales de control de escritura del banco de registros a 0 en el caso de que haya parada.

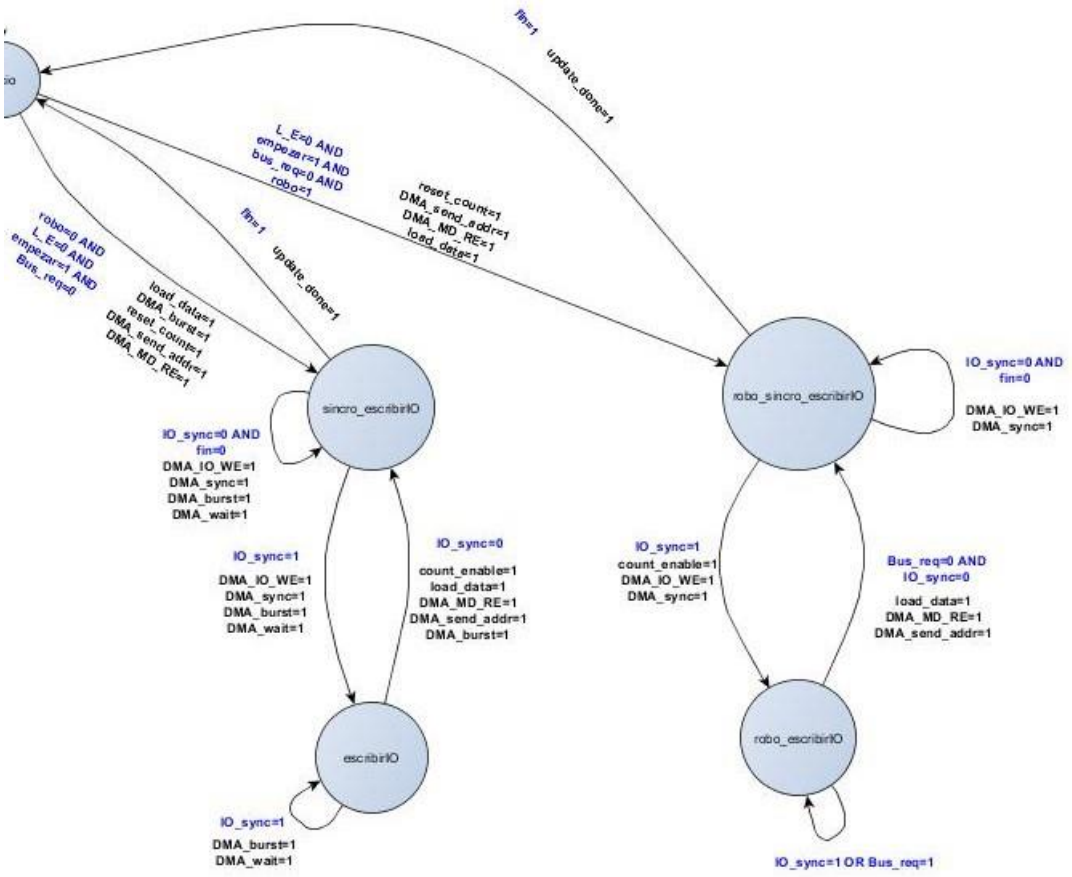
Máquina de estados de la unidad de control del DMA

A continuación, se adjunta el autómata Mealy que programa la UC, dicho autómata consta de 9 estados. Para entenderlo bien se deben tener en cuenta las siguientes consideraciones:

- Las señales de entrada aparecen en color azul.
- Las señales de salida aparecen en color negro.
- De las señales de salida, solo se han puesto las que se ponen a 1 en la transición, las que no aparezcan estarán a 0.
- Se han puesto el autómata dividido en dos partes, primero la izquierda y luego la derecha, ya que si se ponía la imagen global no se podían leer las señales bien.



Nota: En la transición de *sincro_leerIO* a inicio me he confundido con la señal de entrada y es si fin=1.



Explicación de los estados

Inicio

Estado de inicio de la transferencia, se leen las señales de entrada y se salta al estado que corresponda:

- Si nos dicen que se haga con robo de ciclo, que se va a leer de IO y a escribir en MD se salta a *robo_sincro_leerIO* solicitando leer del IO y reseteando el contador de la anterior transferencia.
- Si nos dicen que se haga en modo ráfaga, que se va a leer de IO y a escribir en MD se salta al estado *sicro_leerIO* activando la señal de ráfaga, la señal de wait para que el contador de la MD no empiece a incrementar direcciones a lo loco y reseteando el contador de la transferencia anterior.
- Si nos dicen que se haga en modo ráfaga, que se va a leer de MD y a escribir en IO se salta al estado *sincro_escribirIO*, se lee el dato de MD, se guarda en el registro y se resetea el contador de la transferencia anterior.
- Si nos dicen que se haga con robo de ciclo, que se va a leer de MD y a escribir en IO se salta al estado *robo_sincro_escribirIO*, reseteando el contador de la transferencia anterior, leyendo el dato de MD y cargándolo en el registro de datos, siempre y cuando el bus no lo esté utilizando el MIPS, sino habrá que esperar.

Robo_sincro_leerIO

Vamos a leer de IO y a escribir en MD en modo robo de ciclo, por tanto, hay que hacer petición al IO para que nos dé el dato para guardarlo en el registro.

- Si IO no nos responde se sigue solicitando leer de él.
- Si responde entonces cargamos el dato en el registro de datos y pasamos al estado *robo_leerIO*.

En este estado si nos dicen que la transferencia ha terminado, saltamos al estado de *inicio* a esperar una nueva orden.

Robo_leerIO

En este estado ya se ha cargado el dato en el registro, por tanto, se tienen que bajar las señales de petición de lectura al IO y esperar a que este nos responda bajando su señal de sincronización. Cuando responda, escribimos el dato en MD, incrementamos el contador de la transferencia y saltamos al estado *robo_sincro_leerIO*.

Sincro_leerIO

En este estado, seguimos solicitando leer del IO con la señal de ráfaga activada, la señal de wait y las correspondientes señales de solicitud de lectura. Cuando IO nos responda dándonos el dato, cargamos el dato en el registro, y saltamos al estado *leerIO*.

Si nos dicen que la transferencia ha terminado, saltamos al estado *inicio* esperando a una nueva orden.

LeerIO

En este estado bajamos las señales de petición al IO manteniendo la ráfaga y wait. Esperamos a que IO baje su señal de sincronización, cuando la baje, escribimos el dato en MD, incrementamos el contador de la transferencia y saltamos al estado *sincro_leerIO*.

Sincro_escribirIO

En este estado seguimos solicitando escribir en el IO y esperando a que nos responda para saber que ya ha escrito el dato, manteniendo la ráfaga y wait. Una vez nos responde, saltamos el estado *escribirIO*.

EscribirIO

Es este estado bajamos las señales de petición de escritura del IO y esperamos a que este baje la suya, manteniendo la ráfaga y wait. Cuando IO baje su señal incrementamos el contador de transferencia, leemos un nuevo dato de MD y lo cargamos en el registro de datos por si hay más palabras que transferir, si no hay más, se quedará basura almacenada pero se eliminará cuando se sobrescriba un dato de otra transferencia. Por último saltamos a *sincro_escribirIO*.

Robo_sincro_escribirIO

En este estado solicitamos escribir en IO y esperamos a que este nos responda. Cuando nos responda incrementamos el contador de transferencia y pasamos al estado *robo_escribirIO*. Si nos dicen que se ha terminado la transferencia saltamos al estado de inicio esperando una nueva orden.

Robo_escribirIO

En este estado bajamos las señales de petición de escritura en IO y esperamos a que este baje la suya. Cuando la baje y el bus no lo esté usando el MIPS, leemos el siguiente dato de MD y lo cargamos en el registro por si hay otra palabra que transferir. En caso de que se haya terminado la transferencia, quedará basura almacenada pero se borrará al sobrescribir el dato de una nueva transferencia en el futuro. Por último saltamos a robo*_sincro_escribirIO*.

Pruebas realizadas

Para probar las transferencias en modo ráfaga se ha usado el siguiente banco de pruebas. Básicamente es una modificación del banco de pruebas que se utilizó en el proyecto anterior. Lo que hace el programa es mover una tabla de una zona de memoria a otra introduciendo una transferencia de MD a IO antes de entrar al bucle y dentro del bucle, una transferencia de IO a MD que va desplazándose una dirección de MD por iteración para darle más dinamismo a la prueba y, por último, el programa se queda parado en un bucle infinito de finalización.

dir 0 LW R5, 0(R0) //R0=tamaño tabla

dir 1 LW_pre R30, 20(R2) //Basura en R30 pero nos posicionamos al final de la tabla para empezar a mover

//(Es como hacer un mov a R2)

dir 2 LW_pre R29, 1(R4) //Basura en R29 pero metemos un 1 en R4

//////////////////////INICIO DE UNA TRANSFERENCIA DE MD A IO////////////////////////////

dir 3 LW R6, 48(R0) //Cargamos la orden de la transferencia que se encuentra en la dirección 12 en R6

dir 4 SW R6, 0200(R0) //El MIPS escribe en el registro del DMA para que este inicie la transferencia

//

dir 5 LW_pre R0, 4(R1) //Cargamos el primer dato y movemos cursor

dir 6 NOP R0, 8(R1) //Para engañar al procesador a ver si para

//CONFIGURACION E INICIO DE UNA TRANSFERENCIA CAMBIANTE DE QUE TRANSMITIRÁ //PALABRAS DE IO A MD

///////CAMBIANDO LA DIRECCIÓN DE INICIO DE LA MD POR CADA ITERACIÓN DEL PROGRAMA///

dir 7 LW R6, 44(R7) //Cargamos el dato de la direccion 44

dir 8 ADD R6,R6,R4 //Le sumamos uno para que vaya haciendo

//una transferencia diferente en cada iteración

dir 9 SW R6, 44(R7) //Guardamos el dato actualizado para la siguiente iteración

dir 10 SW R6, 0200(R7) //Iniciamos transferencia

//

dir 11 ADD R3,R3,R4 //Incrementamos el contador

dir 12 SW_pre R0, 4(R2) //Movemos el dato a otro lugar de memoria y movemos cursor

dir 13 BEQ R3, R5, dir11 //Si hemos recorrido toda la tabla, fin del programa

dir 14 BEQ R0, R0, dir5 //Saltamos al inicio del bucle

dir 15 BEQ R0, R0, dir11 //Bucle infinito de finalización

El banco de pruebas para probar las transferencias en modo robo de ciclo se parece mucho al anterior, al principio se utilizó el mismo, pero aparecía un problema: Las transferencias se solapaban, es decir, como en el robo de ciclo el MIPS no se para, se daba la orden de iniciar una transferencia antes de la que estaba ejecutándose hubiera terminado. Por tanto, es tarea del programador encargarse que una transferencia ha terminado antes de dar la orden de iniciar otra.

Se modificó el banco de pruebas anterior para este caso y quedó tal que así.

dir 0 LW R5, 0(R0) //R0=tamaño tabla

dir 1 LW_pre R30, 20(R2) //Basura en R30 pero nos posicionamos al final de la tabla para empezar a mover

//(Es como hacer un mov a R2)

dir 2 LW_pre R29, 1(R4) //Basura en R29 pero metemos un 1 en R4

dir 3 LW R6, 48(R0) //Cargamos la orden de la transferencia que se encuentra en la dirección 12 en R6

dir 4 SW R6, 0200(R0) //El MIPS escribe en el registro del DMA para que //este inicie la transferencia

dir 5 LW_pre R0, 4(R1) //Cargamos el primer dato y movemos cursor

dir 6 NOP R0, 8(R1) //Para engañar al procesador a ver si para

dir 7 ADD R3,R3,R4 //Incrementamos el contador

dir 8 SW_pre R0, 4(R2) //Movemos el dato a otro lugar de memoria y

//movemos cursor

dir 9 BEQ R3, R5, dir11 //Si hemos recorrido toda la tabla, fin del programa

dir 10 BEQ R0, R0, dir5 //Saltamos al inicio del bucle

dir 11 BEQ R0, R0, dir11 //Bucle infinito de finalización

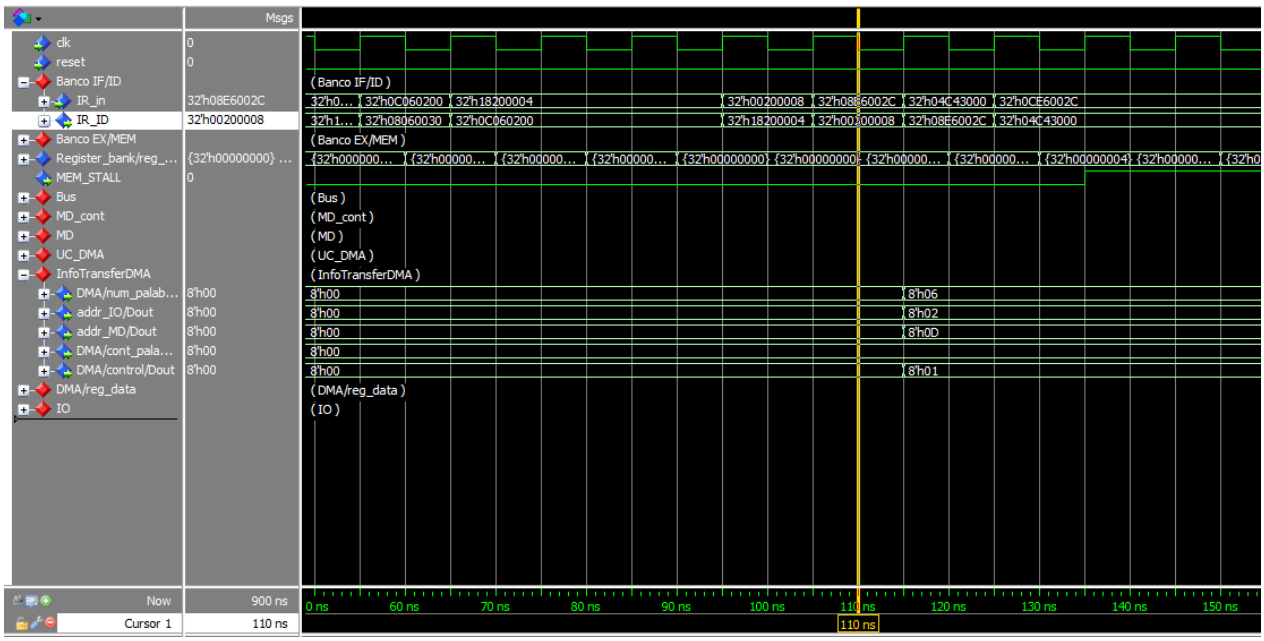
RESULTADOS

Modo ráfaga

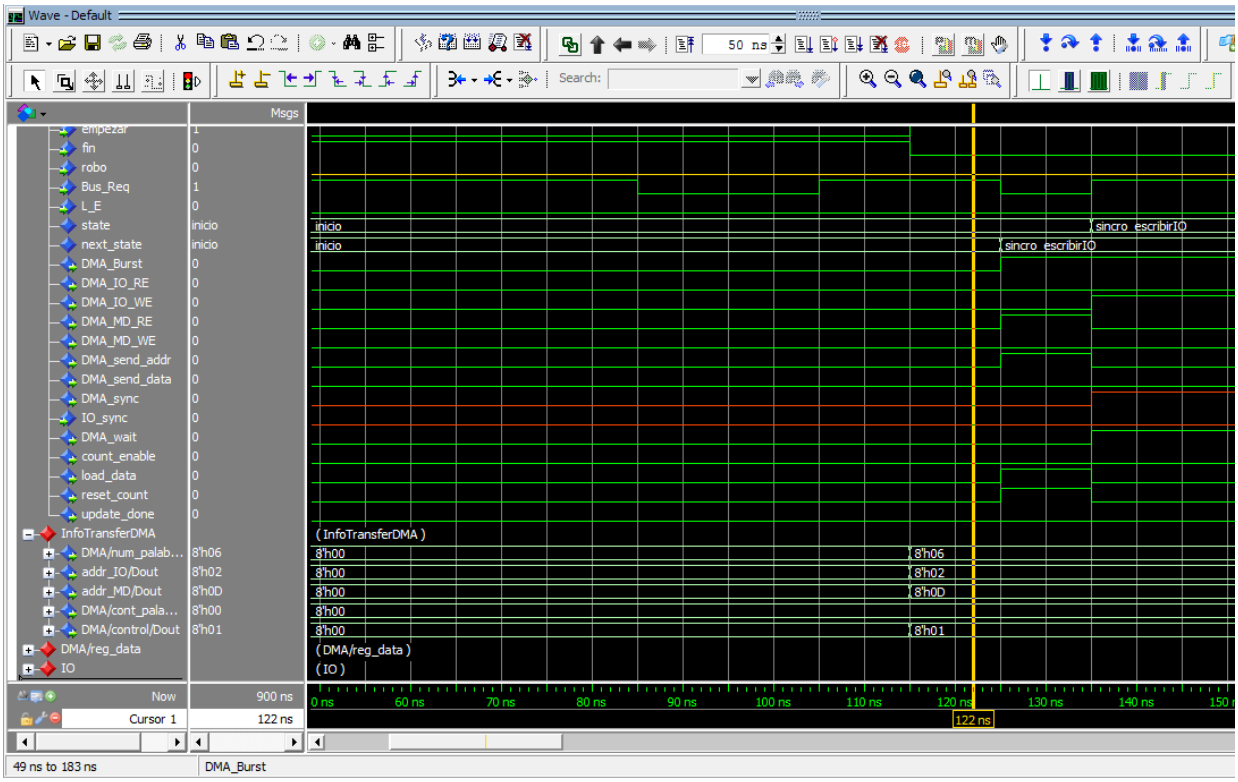
Como resultado de la ejecución el banco de pruebas para el modo ráfaga, se han dado las siguientes órdenes:

- *0106020D*: Mover 6 palabras de la dirección 13 de MD a la dirección 2 de la IO para la transferencia de antes de entrar en el bucle.
- *03061010*: Mover 6 palabras de la dirección 16 de IO a la dirección 16 de MD. Como el banco de pruebas suma 1 a la orden de transferencia antes de almacenarla en el registro de control de DMA, será a la dirección 17 de MD.

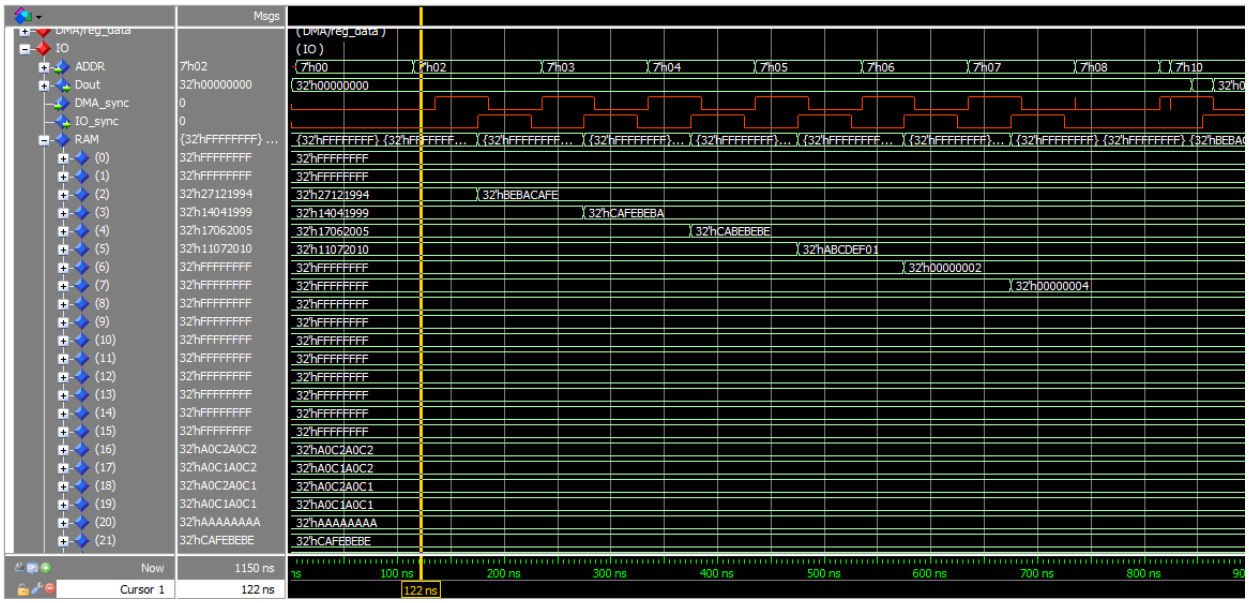
Se puede ver que al ciclo siguiente de que la instrucción que da la orden de transferencia se encuentra en la etapa MEM, la información relativa a la transferencia se ha guardado en el registro del DMA y, poco después, la señal MEM_STALL se activa y en consecuencia el MIPS se para.



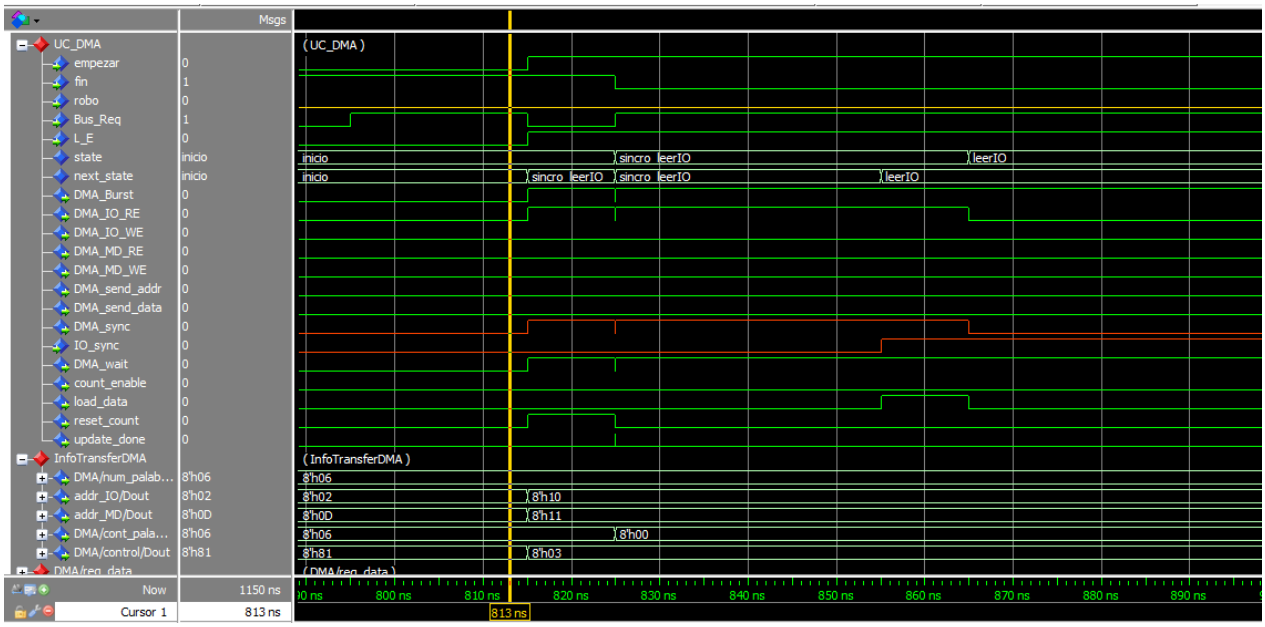
Al siguiente ciclo el autómata se pone en marcha y comienza la transferencia.



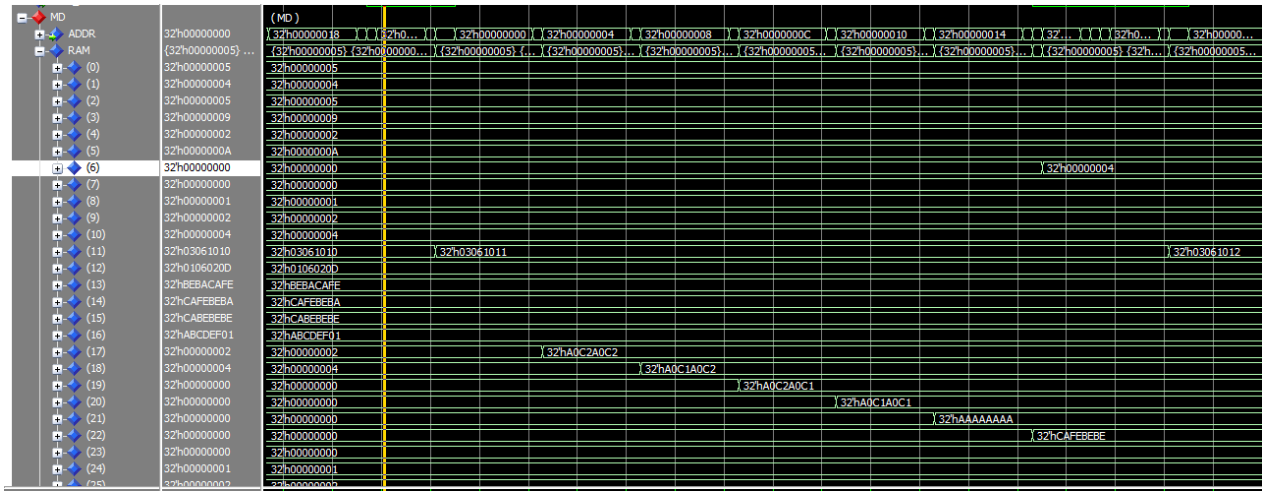
Las palabras se van almacenando en las direcciones de IO según lo previsto.



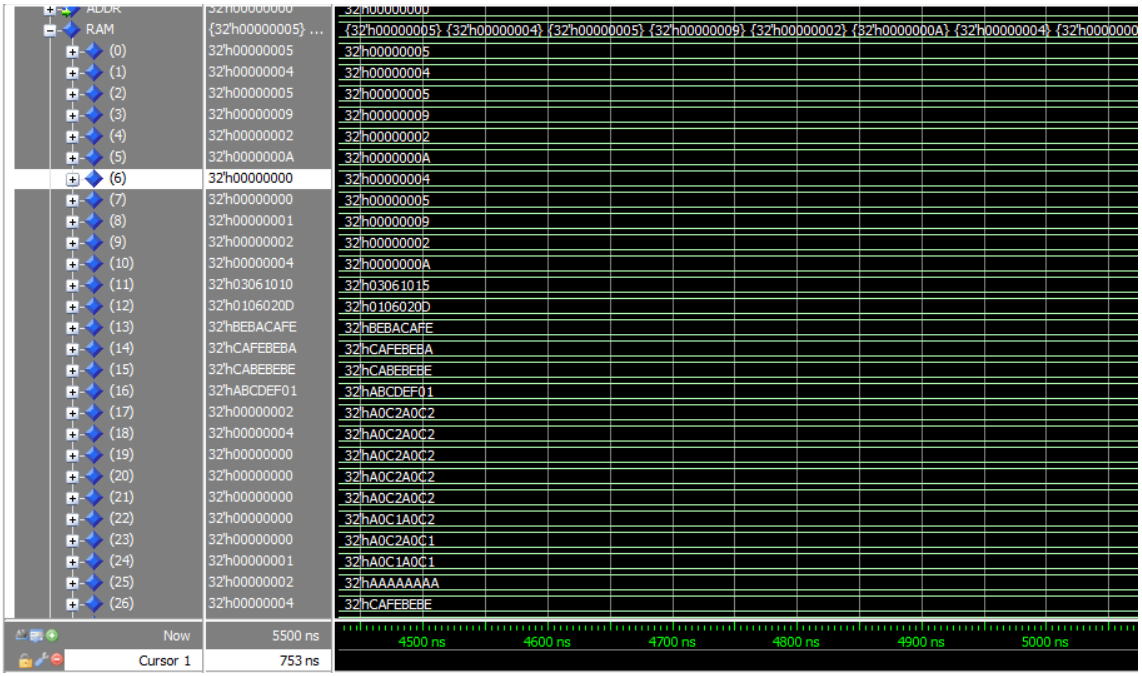
Comienza la primera transferencia de IO a MD



Las palabras se van transfiriendo a MD según lo previsto, el programa sigue ejecutándose cuando la transferencia termina y se incrementa la orden de transferencia para ir recorriendo MD.



Al final de la ejecución el estado de MD es el siguiente.



Podemos ver que hay tantas palabras *AOC2AOC2* como iteraciones tiene el bucle y que la tabla se ha movido entera a la zona de memoria correspondiente.

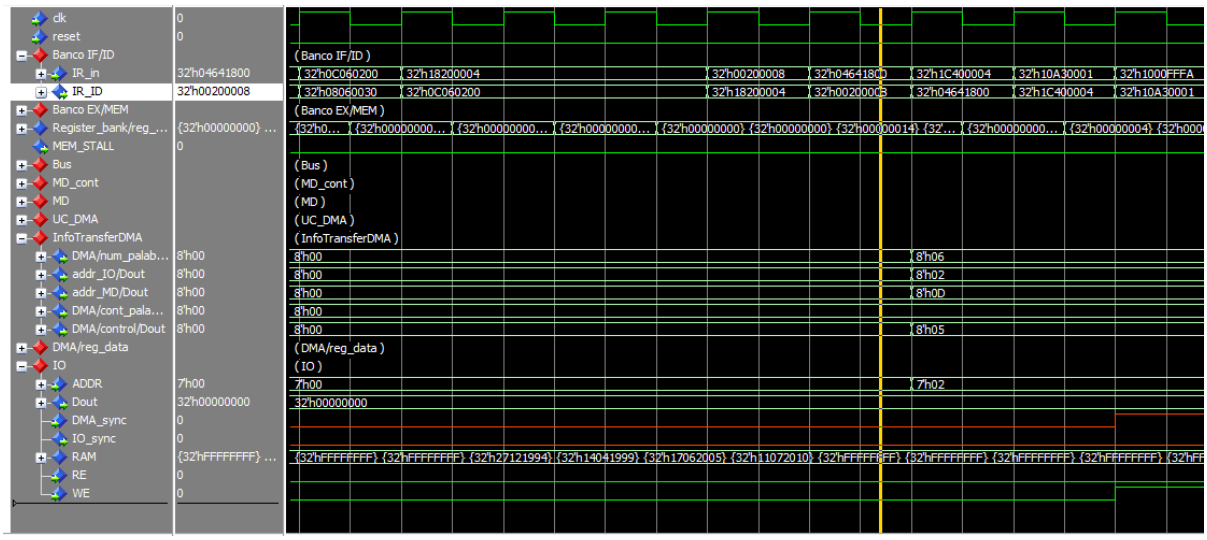
Robo de ciclo

Para el banco de pruebas del modo robo de ciclo, se han realizado dos ejecuciones:

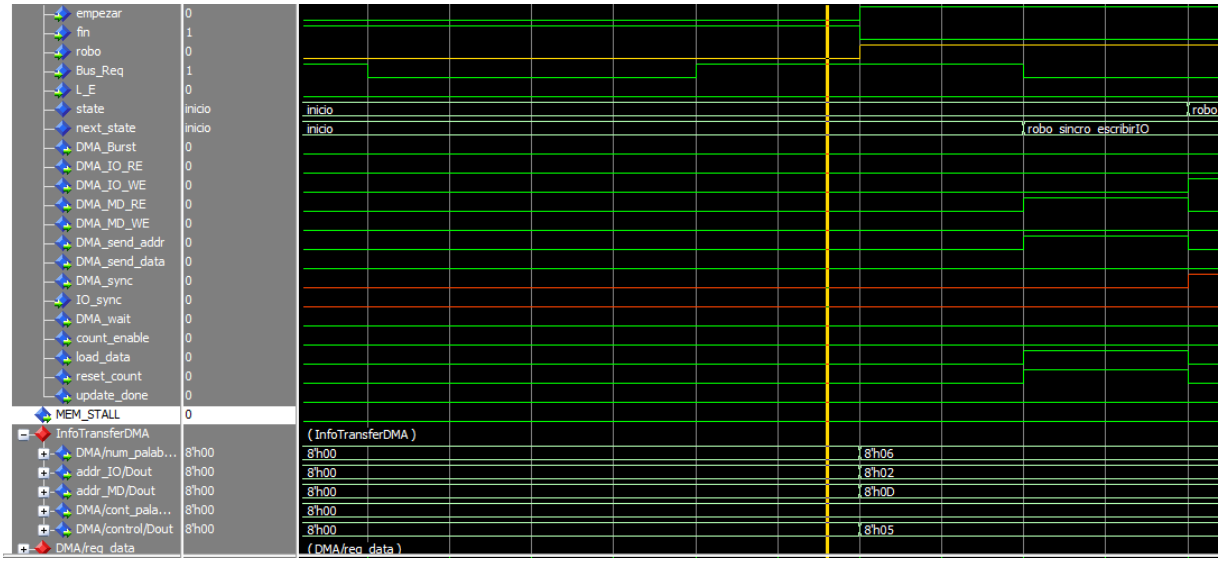
1. *0506020D*: Mover 6 palabras de la dirección 13 de MD a la dirección 2 de IO en modo robo de ciclo.
2. *07061011*: Mover 6 palabras de la dirección 16 de IO a la dirección 17 de MD.

Primera ejecución

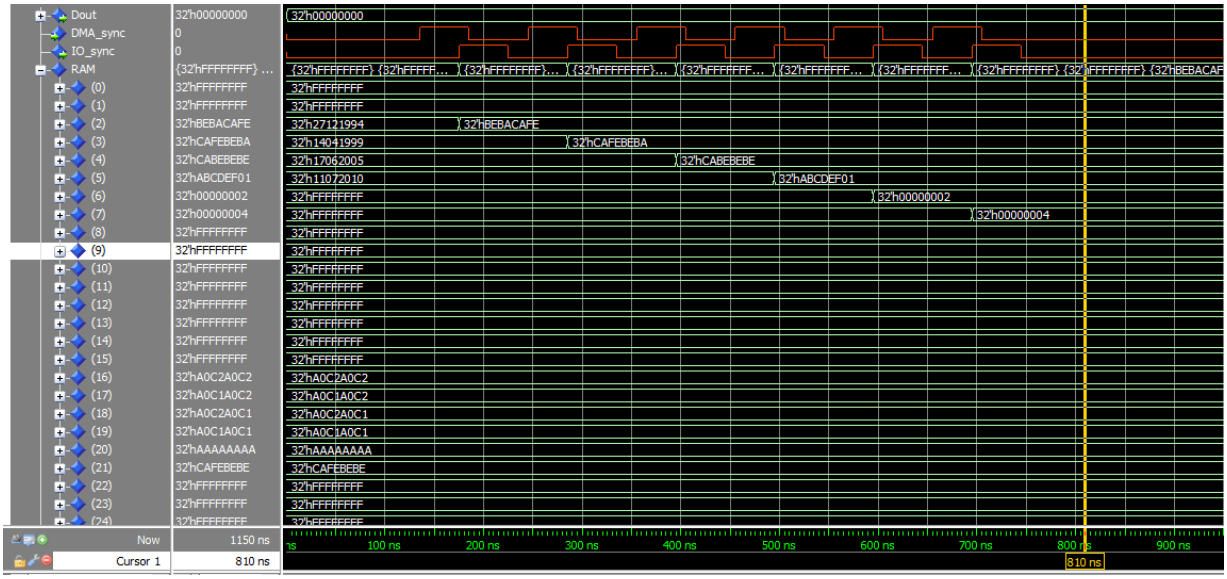
El resultado de la primera ejecución es el siguiente.



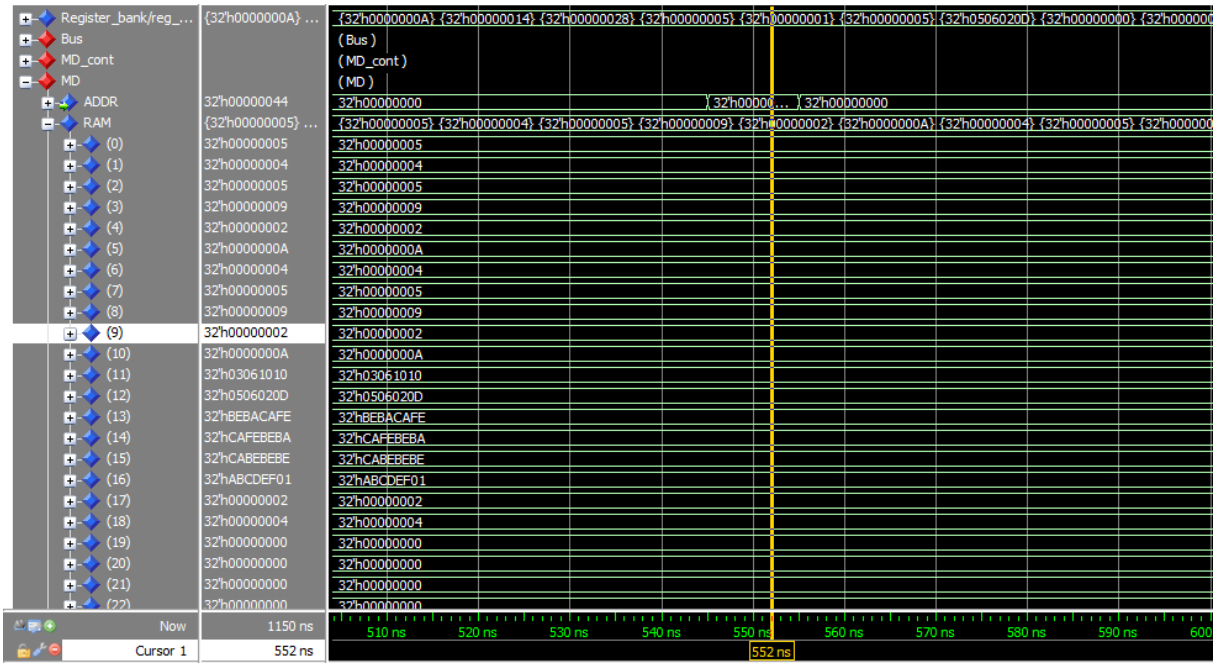
Como podemos ver, los valores de la transferencia se cargan en el registro de control.



La transferencia empieza y la señal MEM_STALL ya no se activa debido a que ya no hay ráfaga.



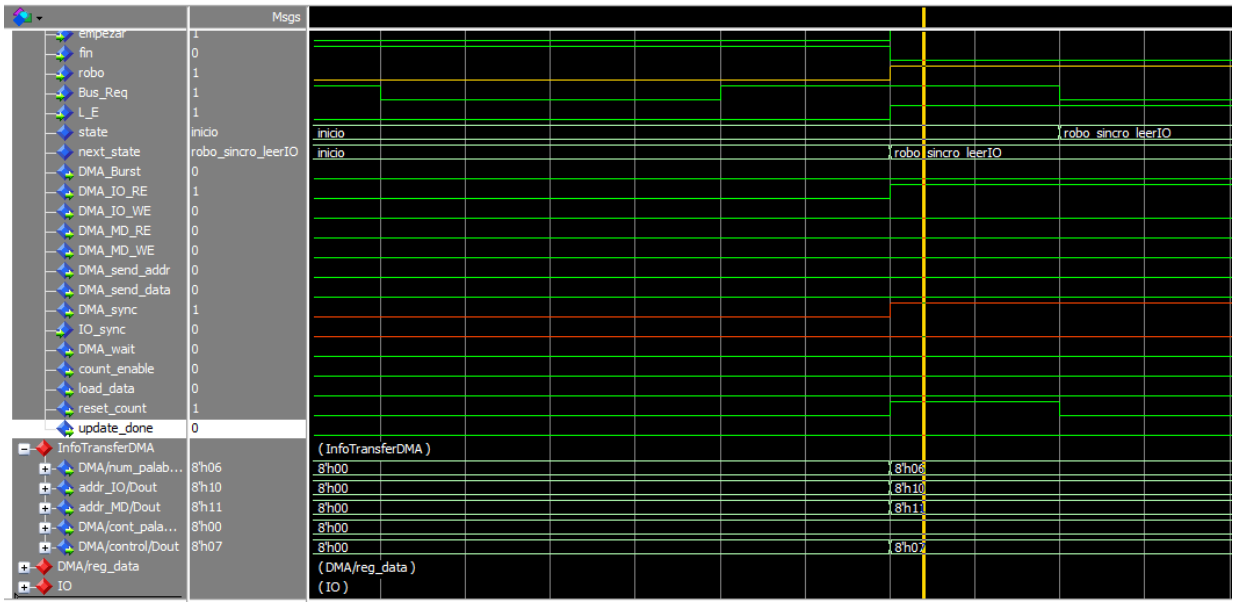
Lo valores se escriben en IO como se esperaba.



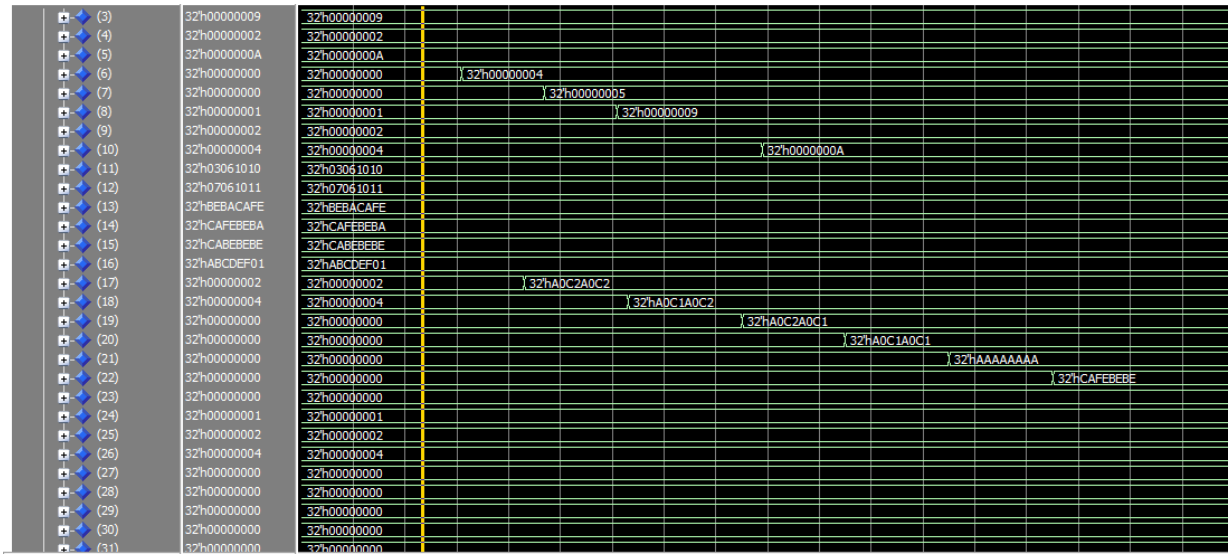
El estado final de la MD es correcto.

Segunda ejecución

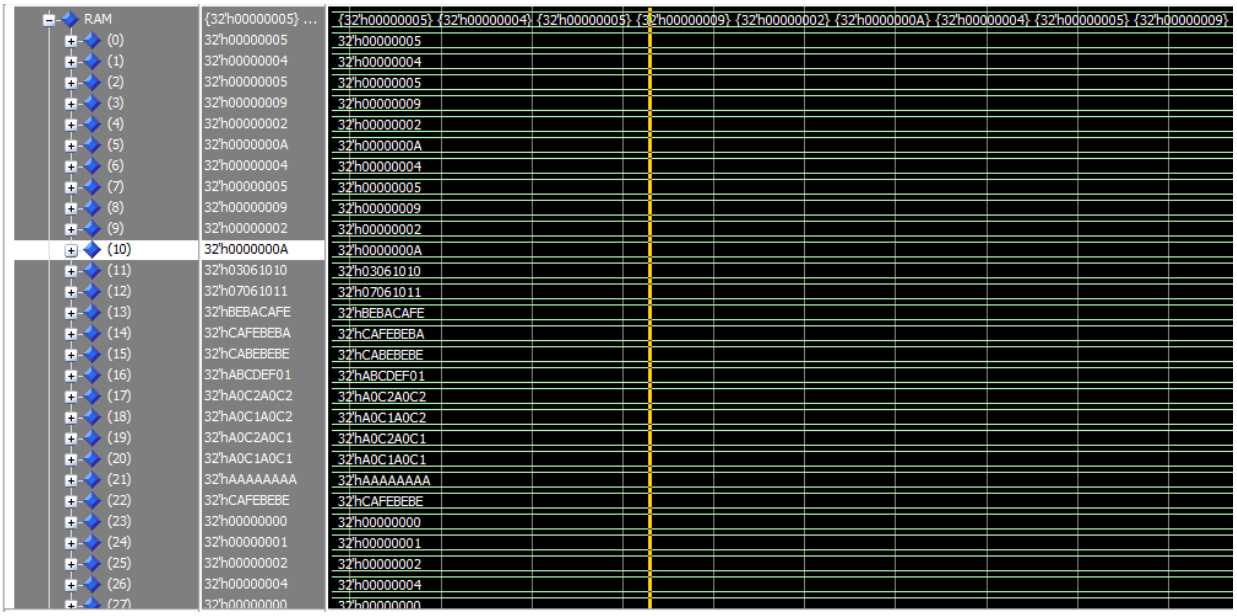
El resultado de la segunda ejecución es el siguiente.



Los valores de la transferencia se cargan en el registro y empieza la transferencia.



La transferencia a MD se hace correctamente mientras que el programa sigue haciendo su ejecución al mismo tiempo. Aclarar que MEM_STALL sigue a 0 toda la transferencia aunque no salga en la imagen.



El estado final de MD es el correcto.

CONCLUSIONES

Se ha aprendido que gestionar un bus es complicado, en este caso solo tenemos un periférico, pero si tuviéramos muchos más la dificultad se dispararía. Las ráfagas son buenas si se tienen protocolos síncronos ya que en pocos ciclos se pueden realizar las transferencias, en caso de tener protocolos asíncronos o semi-síncronos en los que el master solicita permiso y hay que esperar hasta que el slave responda se pierde bastante tiempo de ejecución parando el procesador. Por tanto, se obtienen las siguientes conclusiones:

- Para protocolos síncronos transferencias en modo ráfaga.
- Para protocolos asíncronos o semi-síncronos transferencias en modo robo de ciclo.
- Como se ha hablado anteriormente, en modo robo de ciclo, tener la cautela de que una transferencia ha terminado antes de iniciar la siguiente.

GESTIÓN DE ESFUERZOS

Los esfuerzos del proyecto se encuentran gestionados en la siguiente tabla que se ha ido rellenando progresivamente desde el día en el que se empezó a trabajar.

	Totales	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Estudio de los fuentes	2,5	2,5																	
Diseño inicial de la unidad de control	4,5		1	1,5	1	1													
Depuración y ajustes	28,5						5	4	6,5	2	5,5	5				0,5			
Memoria	7,5													2	3	1,5	1		
Total horas	43																		

Se puede hacer un seguimiento de la evolución del proyecto en el siguiente enlace: https://github.com/danirueda/AOC2-Projects/tree/master/proyecto2