Proyecto evaluación continua 2: Bus compartido y DMA

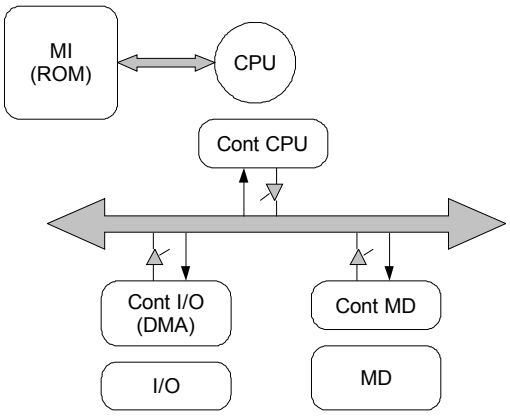
Fecha de entrega: 22 de Mayo

Resumen

En este proyecto debéis diseñar el controlador de bus de un periférico con un controlador DMA que comparte el bus de memoria con el MIPS de la práctica anterior. Debéis gestionar los accesos al bus para que no haya conflictos y las operaciones se realicen correctamente. Además el MIPS puede ordenar al DMA que mueva bloques de datos entre la memoria de datos (MD) y la entrada/salida (IO). Debéis diseñar la unidad de control del DMA para que realice estas transferencias correctamente.

Detalles del sistema a diseñar:

En esta práctica vamos a sustituir la memoria de datos del MIPS por el componente MD_mas_I_O. Este componente incluye un bus y tres controladores de bus. El esquema se puede ver en la siguiente figura:



El bus de memoria:

- **Sincronización**: Es un bus semi-síncrono que soporta ráfagas de tamaño variable. La sincronización se gestiona con las siguientes señales:
    - Si el esclavo no puede realizar la operación en un ciclo activará la señal WAIT y la operación se quedará en espera hasta que WAIT se desactive.
    - Si se quiere realizar una transferencia en modo ráfaga el máster activará la señal BURST al colocar la dirección inicial. Mientras BURST esté activo la operación continuará enviando los datos de direcciones consecutivas. Por ejemplo si el máster pide la palabra 4, el esclavo le dará la 4, la 8, la 12....hasta que el máster baje la señal BURST.
- **Arbitraje**: Tanto el MIPS como el controlador de la I/O pueden actuar como Masters. El arbitraje es distribuido. El bus incluye una línea MIPS_REQ que activa el controlador de la CPU 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.

Controlador CPU:
Siempre actúa como máster del bus. Recibe las solicitudes del MIPS y las traslada al bus. Si la operación es de lectura (lw) le dará el resultado al procesador. Si no puede realizar la operación en un ciclo activa la señal MEM_STALL.
Importante: aparece **un nuevo riesgo estructural** que hay que gestionar. Si el MIPS ejecuta la etapa MEM de un lw o sw y el controlador activa MEM_STALL, el MIPS tendrá que detener

la ejecución (pensad qué etapas deben detenerse) hasta que la operación de la etapa MEM se pueda realizar.

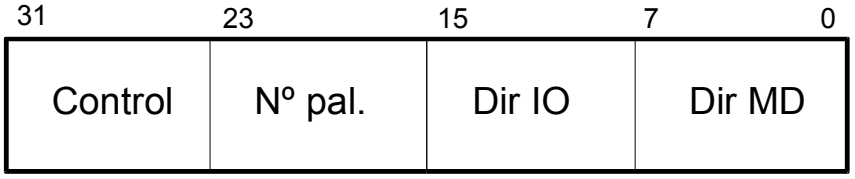
Controlador Memoria de datos:
Siempre actúa como esclavo, trasladando las solicitudes del bus a la Memoria de datos (MD) si están dentro de su rango (X"00000000"-X"000001FF"). La MD siempre realiza la operación solicitada en el ciclo actual. Gestiona las ráfagas con un contador interno que actualiza cada ciclo y resetea al acabar la transferencia. Si el controlador ve que la señal WAIT está activa el contador se para hasta que WAIT se desactive.

Controlador I/O:
Puede actuar como esclavo o como máster. Tiene un interfaz con un registro direccionable por el procesador (con la dirección X"00001000"). Cuando el procesador solicita leer o escribir este registro el controlador responde en el mismo ciclo. Este registro contiene los campos necesarios para programar el DMA para que realice una transferencia de datos entre la IO y la MD. En estas transferencias el controlador actúa como máster. Cuando la IO no pueda realizar su parte, el controlador activará la señal WAIT.

La I/O:
Es un dispositivo de almacenamiento con una memoria similar a la MD pero con una latencia variable. No se puede acceder directamente sino sólo a través del DMA. Para leer/escribir un dato se sigue un protocolo asíncrono utilizando las señales IO_sync y DMA_sync. El DMA es el master y el periférico es el slave. Las palabras se leen/escriben de una en una.

El DMA:
Incluye un registro de control, un contador, un registro de datos y una unidad de control. En el registro de control hay cuatro campos de 8 bits, como se ve en la figura, en los que se indica la dirección del periférico, la de MD, el nº de palabras de 32 bits a transferir, y la información de control. **Las direcciones van en palabras**. Es decir las dirección 1 se refiere a la palabra 1 que está en la dirección X"00000004". De los 8 bits de control sólo se usan 3:

- ***Done*** *(bit 31)* indica que se ha terminado la transferencia anterior. **Lo actualiza el DMA** al terminar una transferencia.
- ***L/E*** *(bit 25)* indica si la operación es de lectura o escritura. Si vale 0 se lee en MD y se escribe en la IO, y si vale 1 es al revés.
- ***Start****(bit 24)* indica que hay que realizar una transferencia. Sólo se le hace caso si ***Done*** tiene un 0.



Por ejemplo si cargamos X"03030603" en reg_DMA el DMA copiará 3 palabras de la dirección 6 de la IO a la dirección 3 de la MD. Y si cargamos X"01021415" copiará 2 palabras de la dirección 21 de memoria a la 20 de la IO.

El contador se usa para saber cuándo se ha terminado la transferencia y para actualizar las direcciones. Por tanto se debe incrementar tras transferir una palabra.

El registro de datos se utiliza como almacenamiento intermedio en las comunicaciones entre MD o IO. En él se guarda el dato leído hasta que se pueda escribir en su destino.

La unidad de control gestiona las comunicaciones del DMA con la MD y con la IO a través del bus compartido. Por un lado debe cumplir el protocolo del bus, y por otro el protocolo de la IO. Se valorará que sea clara y eficiente, es decir que no haya estados redundantes o innecesarios.

Resultados y memoria de la práctica

En primer lugar **debéis comprobar que el diseño funciona correctamente para todos los casos posibles**. Para ello debéis definir esos casos en un banco de pruebas. Os damos ejemplos en los que se prueban algunos casos, pero debéis añadir vosotros mismo el resto de casos que consideréis representativos. **Debéis describir vuestro banco de pruebas en la memoria** explicando qué casos cubre. **Nota importante:** como en la práctica anterior el diseño debe funcionar correctamente para aprobar.

Además, en la memoria debéis explicar brevemente vuestro diseño. Hay que incluir **el diagrama de estados** de la unidad de control, explicando qué se hace en cada estado y qué señales se activan.

Finalmente debéis incluir las **conclusiones** y **tiempo dedicado** por cada componente del equipo.

Apartado optativo

Ampliar la unidad de control para que pueda trabajar en modo robo de ciclo en lugar de en modo ráfaga. Es decir, utilizando el bus sólo cuando está libre, y haciendo transferencias de sólo una palabra. Para ello habrá que enviar siempre la dirección al bus (esto último requerirá un pequeño cambio en el DMA). Es decir si queremos leer cuatro palabras empezando por la cero pediremos la 0, la 4, la 8, y la 16, en lugar de pedir sólo la 0 y mantener la señal BURST a 1 hasta que envíen las cuatro palabras. El bit 26 del registro de control indicará cómo se debe hacer la transferencia. Si vale 1 se trabajará en robo de ciclo.

Este apartado sólo se valorará si se ha realizado un buen banco de pruebas de todo lo anterior. Es mejor hacer una cosa bien, que dos regular.

Anexo 1: ¿cómo incluyo los nuevos fuentes en mi MIPS?

Para incluir la IO hay que sustituir en vuestro MIPS el componente memoriaRAM_D por MD_mas_I_O tanto en la definición como al instanciarlo. También habrá que definir la nueva señal MEM_STALL.

Finalmente incluid todos los fuentes nuevos en el proyecto.

Anexo 2: ¿cómo interactuo desde el Mips con el DMA?

El reg_DMA está mapeado en la dirección x“00000200”. Podemos leer/escribir en el registro haciendo un LW/SW R1, 0200(R0) suponiendo que en R0 haya un 0. Para escribir algo lo más fácil es guardar lo que se quiera escribir en la memoria de datos. Leerlo con un LW, y escribirlo con un SW. Por ejemplo supongamos que en la dirección 0 de la memoria de datos almaceno X"03040102". El siguiente código escribe X"03040102" en reg_DMA:

LW R1, 0(R0)

SW R1, 0200(R0)

Mientras la transferencia se realiza podemos ejecutar un bucle en el mips y comprobar que las instrucciones siguen ejecutándose correctamente. Además leyendo la dirección x“00000200” podemos comprobar si la transferencia ha terminado (si el bit 31 está a 1).

Anexo 3: Estrategia de depuración

Para acelerar la depuración y entender mejor vuestro diseño es importante depurar en cada nivel en el que trabajéis. Es difícil ver los errores de vuestra unidad de control depurando sobre el procesador completo. En lugar de eso hay que hacer un banco de pruebas para cada nivel. Primero comprobad que la máquina de estados funciona cómo pensáis y que genera las salidas correctas. Después comprobar que al unirla al resto del DMA sigue funcionando correctamente. En la siguiente prueba comprobar distintas transferencias en el componente MD_IO. Y sólo cuando hayamos comprobado que funciona lo integraremos en el MIPS. Parece más trabajo, pero ir poco a poco es la clave para depurar eficientemente.

En los fuentes incluimos algunos bancos de ejemplo.

Anexo 4: Estimación del tiempo dedicado

Esta es nuestra estimación:

- Estudio de los fuentes: 2 horas
- Diseño inicial de la unidad de control: 2 horas
- Depuración y ajustes: 16 horas
- Memoria: 3 horas

Cuando nos deis los datos de tiempo dedicado por favor comparaos con esta estimación y analizar las divergencias. La utilidad de vuestro análisis dependerá de que registréis bien los datos. En otras palabras tratad de ser profesionales y no os inventáis un número al acabar la práctica.