Ejercicio controlador DMA en robo de ciclo

Nos piden diseñar un controlador DMA que mueva bloques de memoria principal. El interfaz del DMA tiene cuatro registros direccionables por el procesador:

1. Reg_@_origen
2. Reg_@_destino
3. Reg_num_palabras: indica el número de palabras a transferir.
4. Reg **start** (1bit): el procesador escribe un 1 para ordenar que el DMA comience una transferencia. El DMA lo pone a 0 otra vez al acabar.

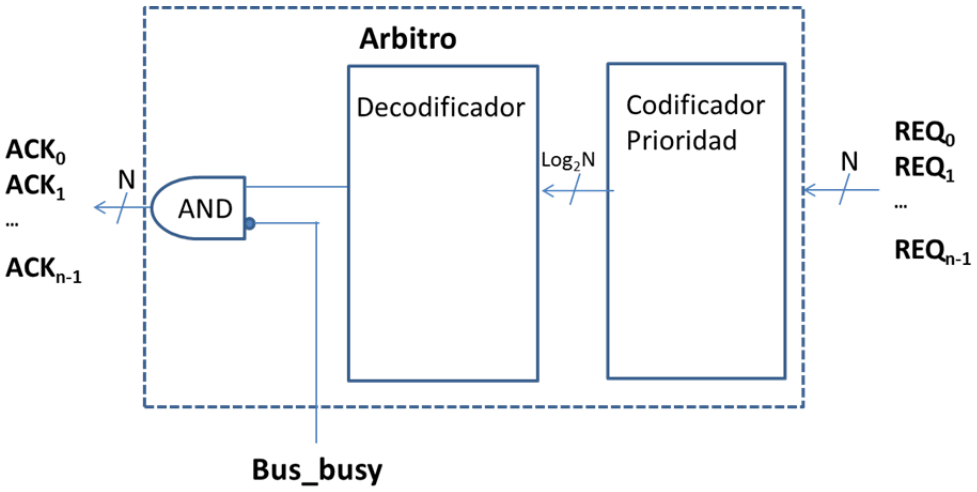
Su funcionamiento es sencillo. El procesador puede escribir los datos de la transferencia en los registros haciendo los "stores" necesarios. Cuando el DMA detecte un 1 en el registro **start** comenzará a leer datos de la dirección de origen y a escribirlo en la dirección destino hasta que haya transferido todas las palabras indicadas.
El DMA funciona en modo **robo de ciclo**. Es decir en lugar de hacer todas las operaciones en una única transferencia del bus (lo que bloquearía el bus durante muchos ciclos), hará cada operación de forma independiente. Es decir, cuando el bus está libre leerá el primer dato y liberará el bus. Cuando vuelva a estar libre, escribirá el primer dato, y volverá a liberar, y así hasta que haya leído y escrito todos los datos.

Detalles del sistema:

El bus de memoria:

- **Arbitraje**: El arbitraje es centralizado y **combinacional**. En la figura se puede ver cómo funciona:
    - Cada dispositivo tiene un REQ y un ACK. Si un dispositivo quiere usar el bus activará su señal **REQ**. El árbitro le avisará de cuándo le concede el bus activando su señal **ACK.** La señal de ACK **puede llegar en el mismo ciclo en que se pidió y es válida sólo para el ciclo en que llega.**
    - Si se necesita utilizar el bus más de un ciclo se activará la señal **BUSY** desde **el ciclo siguiente a recibir el ACK hasta que la transferencia termine**. Esta señal inhibe el arbitraje evitando así que le den el bus a un segundo dispositivo sin que haya terminado el primero. Como se ve en la figura, **si se activa BUSY todos los ACKs del sistema valdrán '0'**.

- **Sincronización**: Es un bus semi-síncrono en el que las operaciones **se pueden hacer enteras en un ciclo** (siempre que el árbitro conceda el bus y el esclavo no active wait):
    - Si el esclavo no puede realizar su parte en el ciclo actual activará la señal **BUS_Wait** y la transferencia se quedará en espera hasta que **BUS_Wait** se desactive.
    - Para trabajar en robo de ciclo se debe garantizar que BUSY valga 0 al menos un ciclo entre cada transferencia, de esta forma el árbitro podrá elegir de nuevo a quién asigna el bus.



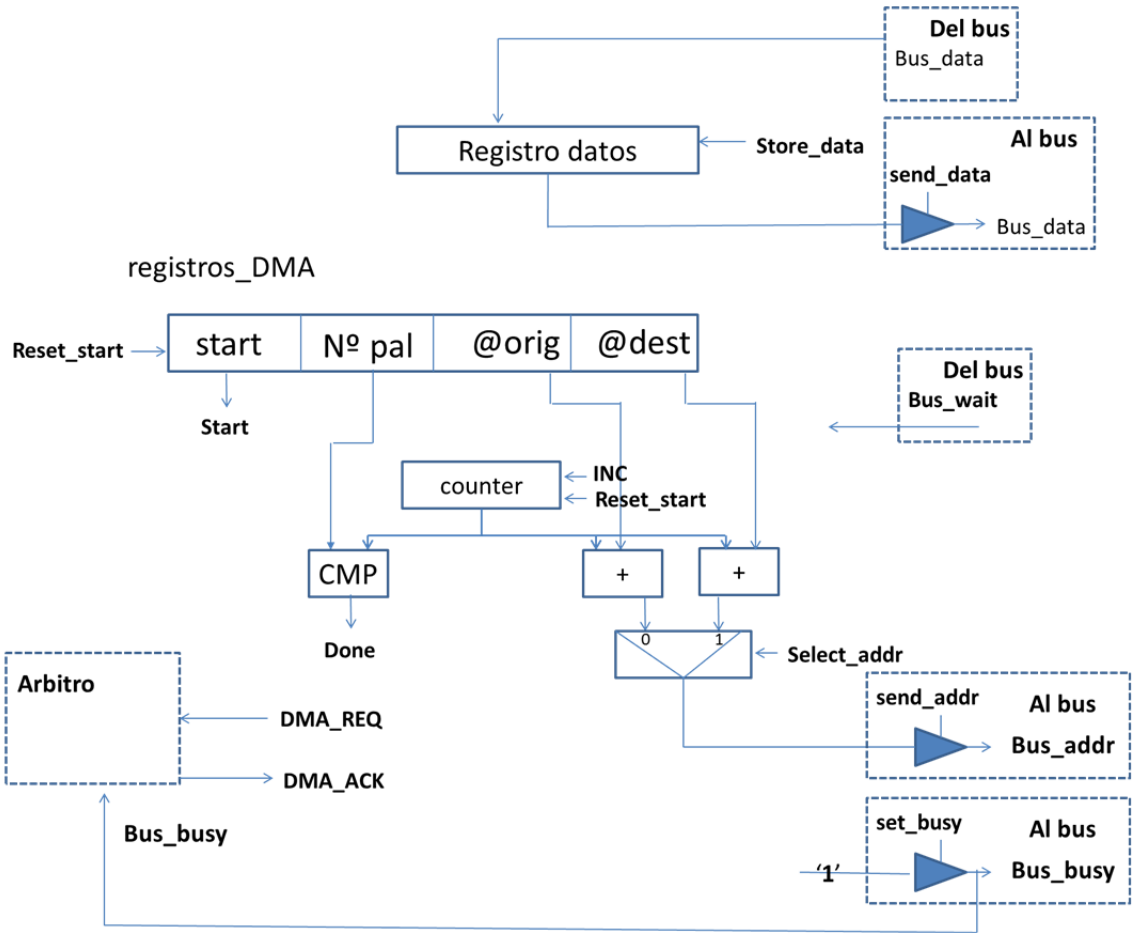
Esquema del árbitro combinacional

El DMA:
En la figura se puede ver la ruta de datos del DMA que además de los registros del interfaz incluye:

- Un contador que lleva la cuenta de cuantas palabras se han transferido. También se usa para calcular la dirección a la que hay que acceder (origen+cuenta, destino+cuenta)).
- Un comparador que avisa cuando se han transferido todas las palabras solicitadas.
- Un registro para almacenar el dato leído de memoria.

Señales a gestionar por el controlador del DMA:

- **Entradas:**
  - **Start:** indica que se debe empezar una transferencia.
  - **Done**: salida del comparador que avisa que ya se han transferido todos los datos solicitados
  - **DMA_ACK**: concesión del bus
  - **BUS_Wait**: la memoria no puede hacer la operación solicitada en el ciclo actual
- **Salidas:**
  - **Reset_start:** pone a 0 el Reg start y el contador.
  - **DMA_REQ**: se activa cuando se quiere solicitar el Bus.
  - **INC**: Incrementa el número de palabras transferidas.
  - **Store_data**: para almacenar el dato leído del bus.
  - **Send_data**: para enviar un dato al bus (también sirve como señal de L/E)
  - **Send_addr**: para enviar una dirección al bus.
  - **Select_addr**: 0 para dirección origen, 1 para dirección destino
  - **Set_busy**: para indicar que el bus está ocupado.



Ruta de datos del DMA