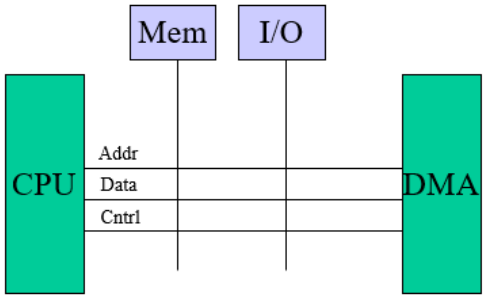
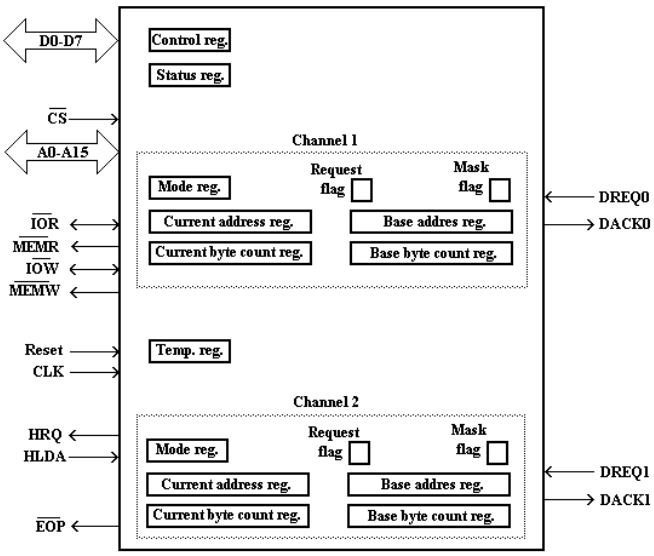
DMA

DMA sta per *Direct Memory Access*, ***è l'unico dispositivo oltre al processore in grado di generare degli indirizzi sul bus per accedere direttamente alla memoria o ai dispositivi mappati in memoria, o direttamente***. Il DMA consente di trasferire grossi blocchi di dati, ad esempio si usa *quando dobbiamo spostare delle pagine di k interi dalla memoria di massa alla memoria centrale*. Con questa caratteristica, a differenza degli altri dispositivi deve porsi al centro delle architetture di bus per potersi sostituire al processore quando c'è da indirizzare un indirizzo di base, di destinazione, capire dove mettere i blocchi di dati, quanti, etc. Sostanzialmente, **il DMA si deve impadronire** del bus, agirà da bus master, permettendo di velocizzare gli accessi in memoria. **Deve gestire l'indirizzamento dei blocchi e i protocolli di comunicazione**, che permettono di sincronizzare i due dispositivi, punto iniziale e finale, in cui trasferire i blocchi stessi. Concettualmente, il DMA *si pone esattamente allo stesso livello del processore*. In figura, se cancelliamo il DMA abbiamo il classico processore che tramite le 3 linee di bus, ovvero data, address e controllo, si collega direttamente a memoria o dispositivi di I/O, sia per gestire il protocollo di scambi di dati sia per indirizzare i dispositivi, per inizializzarli e definire lo scambio. Se vogliamo esplodere la figura, come vediamo a sinistra, vediamo il processore, la cui piedinatura ha dei segnali ancora non considerati, fino ad ora, abbiamo la parte del bus di controllo (in basso) con i classici segnali di lettura e scrittura della memoria. **Il processore Intel, a differenza del 68k, gestisce le periferiche con approccio memory I/O e non memory mapped**. *Nel memory mapped*, come abbiamo visto negli esercizi, *utilizziamo, per trasferire dati dal dispositivo alla memoria, le stesse istruzioni per accedere alla memoria.* Abbiamo le istruzioni di move come le altre istruzioni logico aritmetiche, per fare un'operazione di lettura dalla periferica, spostiamo il dato. *Nelle memory I/O ci sono istruzioni e segnali dedicati per fare l'I/O.* Queste istruzioni e segnali, nel caso 8086, si chiamano **memr e memw, verso la memoria, ior e iow, per le operazioni verso il dispositivo specifico**. Sul control bus verso un qualunque driver abbiamo queste 4 linee, non più 2. Verso l'alto abbiamo altri 3 segnali importanti che permettono di realizzare un **protocollo di comunicazione tra DMA e processore, in cui il DMA chiede al processore il permesso di usare il bus**, ovvero di diventare bus master perché solo così il dispositivo potrà esso stesso scrivere sul bus indirizzi gli indirizzi. Per quanto riguarda i segnali di controllo, sono gli stessi, anche sul DMA avremo memr/w ior/w, per gli indirizzi invece *il DMA prima di poter scrivere sul bus deve chiedere il permesso alla CPU*. Questo avviene tramite un ***protocollo di read bus agreement***, *con i segnali hold request (HLDR) e hold address (HLDA) il DMA richiede al processore la richiesta di bus, ha un ack della richiesta e una volta avuto l'ack può generare gli indirizzi sul bus*, verso ogni tipo di dispositivo. La cosa particolare è il modo con cui il DMA genera gli indirizzi. Esso **può generare linee di indirizzo fino a 16 bit ma**, mentre il processore li genera tutti e 16 insieme, il DMA lo fa in modo particolare, **sono divisi in 3 blocchi differenti, i primi 4 bit servono per indirizzare i registri interni**, 4 perché sono molti registri interni nel DMA, **poi ci sono altri bit che invece vengono montati a partire dal data bus e da un particolare latch, sul DMA, che permette di *staccare fisicamente i segnali che vengono dalla CPU per poter abilitare i segnali provenienti dal DMA stesso***. C'è un altro segnale, infatti, dal processore che si chiama ***bus enabled*** per cui quando otteniamo il bus stacchiamo i segnali del processore e attacchiamo quelli del DMA, delle linee di indirizzo.





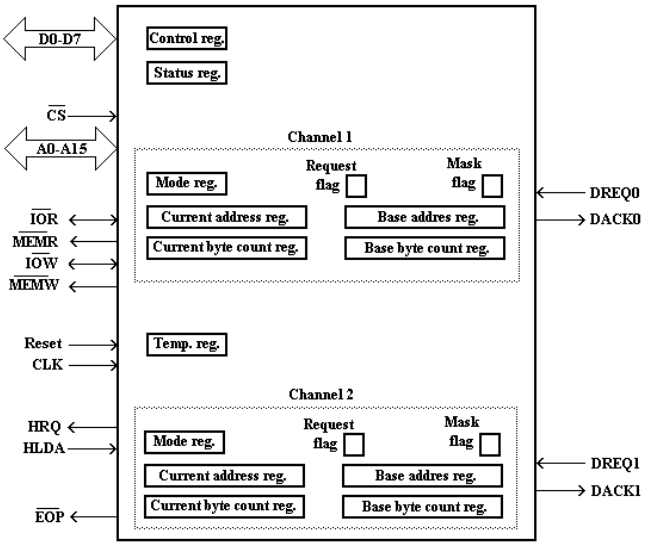
Perché usare un DMA? Quando dobbiamo fare delle operazioni **tra memoria e memoria servono così tante cicli di clock per cui l'operazione è onerosa** per il processore, che non deve fare altro che spostare dati. Si dimostra, ad esempio nel 68k con una frequenza di clock di 500 KHz, riusciamo ad avere una frequenza di trasferimento, senza DMA, di un blocco da memoria a memoria con 50 KHz, quasi un decimo, **col DMA invece riusciamo a sfruttare tutta la freqeunza del clock disponibile**. Per fare la move da memoria a memoria dobbiamo prima fare la fetch dell'istruzione, poi il prelievo del primo dato, poi l'accesso per il secondo dato, il numero di accessi è alto. Nel 68k poi se codifichiamo l'indirizzo di memoria da 32 bit nell'istruzione stessa, anche la fetch da sola richiede molti cicli di lettura di memoria: dobbiamo leggere prima l'istruzione, poi il primo indirizzo, che è da 32 bit (2 accessi), poi il secondo (2 accessi), sono 5 word quindi 5 accessi in tutto.

Architettura del dispositivo

Il dispositivo DMA è fatto da tante sezioni differenti dette ***canali***, ogni canale va programmato separatamente ed ha, separatamente, i suoi registri interni e le linee di collegamento verso un dispositivo particolare. **Ad ogni canale può essere associato un device**. Poi abbiamo dei **registri generali, ovvero il registro di controllo e di stato** (che valgono per tutti) **e per ogni canale abbiamo un generatore di indirizzi di base e un contatore**. **Per generare gli indirizzi si utilizza un contatore**, quando dobbiamo trasferire un blocco di dati, tipicamente questo si trova a indirizzi consecutivi. Indichiamo allora di trasferire da un determinato indirizzo i 16, ad esempio, byte successivi: **tramite un contatore e gli indirizzi base gestiamo tutti gli indirizzi del blocco da trasferire**. Ci sono delle operazioni di inizializzazione che ci permettono di indirizzare, dare l'indirizzo base del blocco da trasferire. Abbiamo dei registri temporanei, all'interno del canale, che ci permettono di memorizzare temporaneamente un blocco in transito. Ci sono poi tutta una serie di segnali che servono per stabilire il protocollo di comunicazione con il dispositivo e una serie di segnali per "rubare" il bus al processore, il DMA si dice lavori per "***cycle stealing***", rubando cicli del processore. Vede quando il processore è inattivo sul bus per rubargli il bus. **Quando il processore cede il bus può continuare internamente a lavorare, può fare tutto tranne le operazioni che richiedono l'accesso alla memoria esterna**, se dobbiamo fare la somma tra due registri interni il bus non serve e lo possiamo cedere a un altro dispositivo che per noi trasferisce blocchi. Il nostro riferimento è il dispositivo Intel 8237, dalla sua architettura dobbiamo considerare come prima cosa che abbiamo 4 canali di collegamento con 4 dispositivi, in Asim invece sono solo 2. Poi, **ci potrebbe essere un problema di competizione tra i canali, non hanno ognuno un bus dedicato ma devono conquistare il bus unico**, si deve gestire un meccanismo di priorità, all'interno del DMA c'è un blocco che si occupa della gestione della priorità e potremo programmare i canali affinchè si possa avere **o una priorità fissa**, ad esempio il canale 3 ha priorità maggiore della 0, **in funzione della posizione del canale, oppure usiamo una priorità dinamica, con un round robin**, la priorità gira e in tal modo un canale non può tenere sempre per sé il bus, questo è preferibile e Intel permette entrambe le modalità. Per accedere al bus, al canale dati, possiamo farlo tramite **4 modalità diverse di indirizzamento**, le prime due sono single o block, rappresentano per quanto tempo rubiamo il bus:

- Nella ***modalità single*** si trasferisce una parola alla volta, **dopo aver trasferito la parola il DMA restituisce il bus al processore**, almeno per un ciclo di bus. Trasferiamo un blocco alla volta e restituiamo il controllo del bus al processore, non sempre è utile ma se non abbiamo molti blocchi da trasferire può avere senso.
- Una modalità più interessante è la ***modalità block***, in cui si trasferisce un intero blocco non appena il DMA ha acquisito il controllo sul bus, **alla fine del trasferimento dell'intero blocco il DMA invia un segnale di interruzione al processore che indica fine trasferimento**. La programmazione del DMA rispetto ai dispositivi visti è la più semplice, ha una sola linea di interruzione che dice solo ho finito, il DMA fa tutto da solo quindi è semplice da programmare.
- Ci sono altre due modalità, meno interessanti e meno critiche, che non possiamo realizzare su Asim. La ***modalità on-demand*** è simile alla block ma con la differenza che il trasferimento del blocco continua fintanto che la linea di richiesta è attiva ma **il trasferimento può essere sospeso, dal processore, per poi essere ripreso esattamente dal punto in cui è stato sospeso**. Dunque, si prevede la possibilità che il processore interrompa il blocco. Grazie ai registri contatore, che salvano anche il conteggio corrente, siamo in grado di riprendere il trasferimento da dove è stato interrotto.
- La ***modalità cascade*** permette di **collegare più DMA tra loro**, come visto con il PIC, per gestire contemporaneamente più di 4 canali.

In figura vediamo l'architettura interna, semplificata con Asim. Asim prevede da un parte dei registri generali che servono per programmare il dispositivo, poi ci sono dei registri specifici per programmare i singoli canali. Abbiamo dei **registri di controllo, di stato e temporanei, comuni ad ogni canale**, e poi abbiamo 4 canali. **Per ogni canale avremo un registro di modo, due registri indirizzi** che rappresentano indirizzo base e corrente, **due contatori** che rappresentano il contatore base e il valore del contatore corrente, poi ci sono una serie di flag che funzionano da **flag di richiesta** per attivare le interruzioni e da **flag di mascheramento** per gestire le priorità. I segnali sono notevoli per ogni canale, nella parte a destra in figura abbiamo *dei segnali che sono associati al singolo canale e permettono a un dispositivo di fare una richiesta di bus*. **Abbiamo le coppie request/aknowledge per ogni canale**, poi per l'interfacciamento di scrittura e lettura abbiamo i 4 segnali **memr/w ior/w**, poi abbiamo il **clock**, una **linea di interruzione EOP**, e una coppia di linee per gestire il protocollo sul bus, ovvero **HLDR e HLDA** (hold request/ack). Abbiamo 4 segnali per read/write e **abbiamo un circuito dedicato che converte i segnali di DMA in quelli reali della memoria**. Notiamo che ***EOP è bidirezionale*** e richiede un'interruzione per indicare il trasferimento completato. EOP è un segnale di interruzione ma



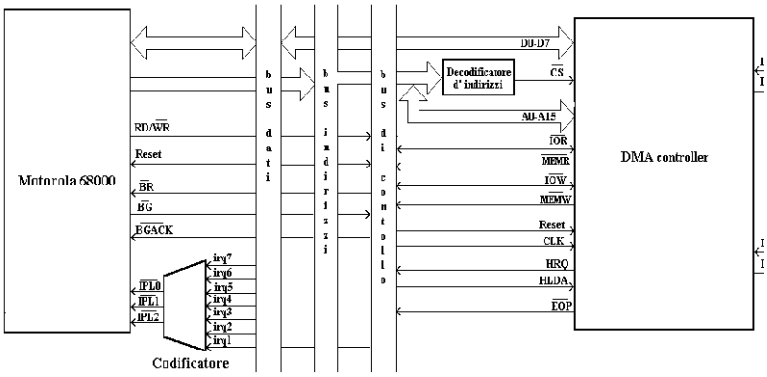
è bidirezionale, a differenza di quelli visti, ***non solo il DMA manda un'interruzione al processore per dire che ha finito ma può succedere il contrario, il processore interrompe il DMA*** e per farlo manda una segnale proprio su questa linea.

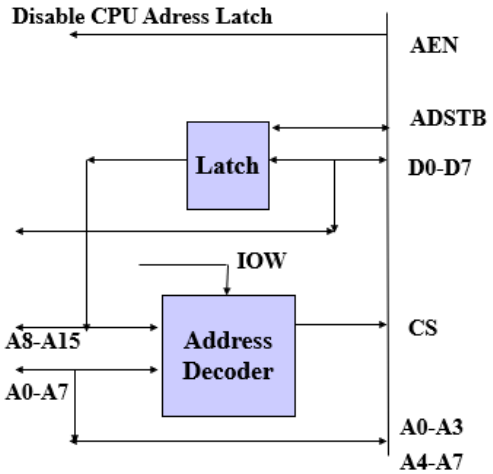
Protocollo per trasferire dati

Vediamo il protocollo per trasferire dati da device a memoria o da memoria a memoria. Per prima cosa ***il DMA deve richiedere il bus al processore o in generale al bus master*** (che noi diamo per scontato che sia il processore). Abbiamo un segnale di bus request e appena viene ricevuto il bus grant, o hold ack per la nomenclatura Intel, **il DMA pone l'indirizzo base sull'address bus**, manda un ack al device che lo aveva richiesto**, abilita i segnali di ior memw perché dobbiamo leggere dal dispositivo e scrivere in memoria** (caso device-memoria), il dispositivo pone i suoi dati sul bus e disabilita la richiesta al DMA, la memoria legge i dati e **manda un'operazione di ready al controller che, su quel canale, incrementa l'indirizzo corrente e decrementa il contatore di byte**, il ciclo continua fintanto che il contatore non viene decrementato fino allo 0, appena finisce il processore si invia un'interruzione al processore o al dispositivo per indicare che il trasferimento è terminato. **Se l'operazione è di read da memoria e write sul device abilitiamo iow e memr**.

Interfaccia

Questa in figura è l'interfaccia che vedremo con il 68k che **non ha tutti questi segnali perché il 68k è memory mapped, non ha ior/w memr/w**, bisogna interfacciarsi con i segnali relativi al processore. **Non abbiamo i segnali HLDR e HLDA ma bus request e bus grant e bus grant ack, il protocollo per richiedere il bus anche è leggermente differente**. L'Intel può lavorare in due modalità, la ***modalità slave*** è la *fase di inizializzazione, per accedere al dispositivo e ai registri interni per inizializzarlo, definire le priorità e il canale per il trasferimento dei dati*. Si può vedere come una periferica con la differenza che **usa 16 bit per accedere alla periferica, i primi 12 identificano la periferica vera e propria, gli ultimi 4 fanno accedere ai registri interni con 4 bit**, abbiamo 2^4 possibili indirizzi differenti per accedere ai vari canali che devono essere inizializzati. Nella ***modalità master*** invece succede qualcosa di diverso, *il DMA fa da master del bus.* Questo significa che **c'è un problema elettrico, si devono disconnettere le linee di indirizzamento collegate al processore per abilitare il DMA a scrivere anche su queste linee**. Il protocollo per l'architettura e scrivere le linee di indirizzo sul bus è un poco più complesso. Quando il dispositivo funziona da master deve generare gli indirizzi***, quando è attivo il segnale AEN, di enable, disabilita il latch di indirizzi provenienti dal processore. L'indirizzo su 16 bit viene formato mettendo gli 8 bit meno significativi dell'indirizzo sui pin a0..a7 del dispositivo, invece gli 8 più significativi*** non sono messi sulle linee di indirizzo ma ***sono messi sulle linee dati***, tramite un segnale di ***address strobe*** viene abilitata la scrittura sull'address register, i 4 bit di completamento dell'indirizzo a16..a19 devono essere programmati a priori (non sono generati dal DMA) e non possono essere modificati durante il trasferimento dei blocchi. Inoltre, genererà i segnali di lettura e scrittura, sia verso il dispositivo (IOR/IOW) sia verso la memoria (MEMR/MEMW) che dovranno essere tradotti con un opportuno circuito di codifica, per garantire la piena compatibilità con i segnali di accesso alla memoria o al dispositivo.





Graficamente, l'indirizzo da a0 ad a16 è costruito da due parti, la prima proveniente dai primi 8 bit sul data bus del DMA (che a destra non si vede), **gli indirizzi a8 a15 invece si costruiscono a partire dagli 8 bit del bus dati del DMA quando con l'address strobe disabilitiamo gli indirizzi del processore, tramite il latch, e abilitiamo quelli del DMA**. Il trasferimento da memoria a memoria è possibile e usato quando vogliamo trasferire blocchi di memoria. In questo caso, il byte del blocco da trasferire **viene portato dalla sorgente a un registro temporaneo interno al DMA e poi da questo viene trasferito all'indirizzo di destinazione della memoria stessa**. In questo caso saranno necessari **2 cicli e 2 canali** perché nel primo canale indichiamo la sorgente, nel secondo invece indichiamo la destinazione. Questo non avviene quando dobbiamo fare un trasferimento di tipo memoria-device.

Abbiamo una serie di registri generali di controllo, di stato e poi di modo, nel ***mode register*** tratteremo informazioni relative, ad esempio, a *se vogliamo usare un conteggio a incremento o decremento, se vogliamo abilitare o meno l'autoinizializzazione dell'address, per ogni canale*. Nel ***registro di controllo*** definiamo *se vogliamo fare un trasferimento memoria memoria (2 canali) o device-memoria*, possiamo specificare la priorità dei canali e via dicendo, poi **abbiamo sia lo status register che i flag che permettono di gestire protocolli e interruzion**i. Ad ogni canali c'è un ***request flag***, una richiesta

del DMA, è l'equivalente di una richiesta al DMA da un device tramite il segnale ufficiale, diretta. Questa richiesta in Asim la possiamo programmare sia in software sia gestirla quando vogliamo fare una richiesta hardware. **Quando il DMA finisce e alza EOP verso il processore, il flag si azzera. Il flag di maschera invece permette di definire le priorità**, disabilitiamo il canale da ogni richiesta, come se lo mascherassimo.

Dobbiamo gestire sia il protocollo verso il bus del processore, per diventare bus master, sia le richieste verso i dispositivi che vogliono interrompere. Possiamo trovare tutto questo sul datasheet, possiamo vedere i protocolli nel dettaglio, il protocollo di comunicazione e il meccanismo per gestire e generare l'indirizzo da utilizzare sull'address bus.