Corso di *Architettura dei Calcolatori* 2023-2024
Realizzazione di DMA Controller 8237A in *Verilog* (Traccia A01) – Relazione Finale

Partecipanti: Francesco Interlandi (f.interlandi@student.unisi.it), Lorenzo Nigi (l.nigi3@student.unisi.it), Vittorio Sali (v.sali@student.unisi.it)

Introduzione

Abstract

In un PC il ruolo del DMA Controller è quello di gestire i flussi di dati tra CPU e periferiche I/O e memorie. Inizialmente la CPU imposterà il DMAC (che quindi si comporterà da *Target*) tramite i segnali di ingresso, specificando indirizzi iniziali, indirizzi di destinazione, direzione e quantità di dati trasmessi, oltre che un segnale di start. Successivamente il DMAC si comporterà da *Initiator* e genererà i segnali necessari al controllo del trasferimento dei dati; al termine dell'operazione invierà un interrupt alla CPU.

Per la realizzazione del modulo *Verilog* è stata usata una rappresentazione del DMAC come una rete sincrona (rispetto ad un clock fornito dalla CPU), tramite FSM, modello di Mealy, dato che la variazione dello stato viene valutata ad ogni ciclo di clock, in funzione sia degli ingressi che dello stato corrente.

Modalità di lavoro

Vista la necessità di una visione d'insieme del funzionamento del DMAC, il quale è risultato essere un dispositivo complesso e operante tramite molti processi dipendenti l'uno dall'altro, principalmente è stato scelto di lavorare collettivamente, tramite l'uso di un editor di testo condiviso, in modo da poter modificare il codice in contemporanea, da diversi terminali.

Risorse utilizzate

- "c124lez16-dma_extended.pdf", Roberto Giorgi, Università di Siena, C124L16, slides fornite dal Professore presso il sito: https://arcal.dii.unisi.it/tools1/
- Intel 8237A Datasheet: 8237A HIGH PERFORMANCE PROGRAMMABLE DMA CONTROLLER (8237A-5), file "8237A.pdf" fornito dal Professore
- https://www.mouser.com/datasheet/2/225/MultichannelDMAControllerUsersGuide-1670817.pdf , come riferimento per la realizzazione della Finite-State Machine, modello di Mealy, del DMAC.
- Compilatore *Verilog*, "*Icarus Verilog*": https://github.com/steveicarus/iverilog
- Visualizzatore per forme d'onda "*GTKWave*": https://gtkwave.sourceforge.net/

Limitazioni del programma realizzato

Non sono state implementate le modalità operative del DMA per il trasferimento Memory-to-Memory e la modalità Dummy Transfer.
Inoltre, per questioni di complessità di sincronizzazione dei trasferimenti, non è stato realizzato un TestBench

per la modalità Cascade, anche se ne è stata comunque fornita un'implementazione, che sarà descritta a seguire.

Nella realizzazione presentata non sono stati implementati i ritardi generati dalle reti combinatorie, necessarie per calcolare le espressioni condizionali richieste, tuttavia, scegliendo un periodo per il clock del DMA sufficientemente esteso, è possibile ovviare a tale limitazione.

La sintesi del codice non è possibile usando il programma *Verilogger*, dato l'elevato numero di righe che compongono il codice; tuttavia, è possibile utilizzare il programma open-source *Icarus Verilog* per la sintesi del codice e per la generazione del file di uscita contenente le forme d'onda, che può essere poi analizzato tramite il programma *GTKWave*, anch'esso open-source.

Sezioni

- Modulo DMAC
- Modulo Priority Encoder
- Modulo TestBench

DMAC

Porte input/output

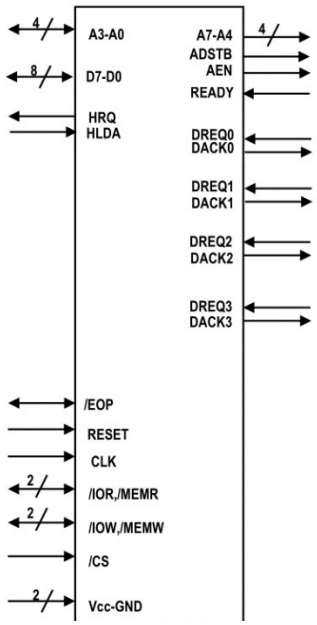

Fig. 1: Rappresentazione simbolica del Chip DMAC (da c124lez16-dma_extended.pdf, slide 25)

- **inout[3:0] A30:** In input sono utilizzati dalla CPU per selezionare i registri che si vogliono leggere o programmare. In output forniscono i 4 bit meno significativi dell'indirizzo in uscita.
- **input A30_oe:** *Output enable* (per il DMAC), usato per la configurazione a Tri-State Buffer della porta I/O A30.
- **inout[7:0] D:** Viene utilizzato per lo scambio dei dati su 8 bit. Nel caso in cui AEN assume valore alto, i bit di D7-D0 sono utilizzati come A15-A8.
- **input D_oe:** *Output enable* (per il DMAC), usato per la configurazione a Tri-State Buffer della porta I/O D.
- **output HRQ:** (Hold ReQuest) Viene utilizzato dal DMAC per richiedere al processore l'accesso al BUS.
- **input HLDA:** (HoLD Acknowledge) Permette di segnalare che il processore ha ceduto l'utilizzo del BUS (al DMAC).
- **inout EOP_:** (End Of Process) In input segnala che è stata effettuata dall'esterno una richiesta di terminare le operazioni mentre; in output, indica la presenza o meno di una richiesta interna di terminare le operazioni.
- **input EOP_oe:** *Output enable* (per il DMAC), usato per la configurazione a Tri-State Buffer della porta I/O EOP_ (in generale, durante la modalità operativa avrà valore alto, ovvero il DMAC sarà in grado di scegliere arbitrariamente quando inviare un *End Of Process*).
- **input CLK:** Sincronizza le operazioni interne di 8237A e la sua velocità di trasferimento dati.
- **inout[1:0] IOR_MEMR_:** Lettura da/verso I/O e/o da/verso memoria.
- **input IOR_MEMR_oe:** *Output enable* (per il DMAC), usato per la configurazione a Tri-State Buffer della porta I/O IOR_MEMR_.
- **inout[1:0] IOW_MEMW_:** Scrittura da/verso I/O e/o da/verso memoria.

- **input IOW_MEMW_oe:** *Output enable* (per il DMAC), usato per la configurazione a Tri-State Buffer della porta I/O IOW_MEMW_.
- **input READY:** attivato dal target: indica che il target è pronto per l'inizio del trasferimento (analogo ad un segnale WAIT_)
- **input CS_:** (Chip Select) Viene utilizzato per l'abilitazione del chip.
- **input[1:0] Vcc_GND:** È utilizzato per l'alimentazione del dispositivo (su queste porte non avviene scambio di dati).
- **output[3:0] A74:** Fornisce i 4 bit più alti della parte meno significativa dell'indirizzo di uscita, cioè quella formata dai bit A74-A30.
- **output ADSTB:** (ADdress STroBe) Memorizza su un latch esterno i bit A15-A8, cioè D7-D0 quando AEN assume valore alto.
- **output AEN: (**Address ENable) abilita l'uso di D7-D0 come A15-A8 per permettere al sistema di utilizzare 16 bit per l'indirizzamento.
- **input DREQx (x = 3-0):** Le *DMA Request Lines* vengono utilizzate ognuna dalla rispettiva periferica i-esima per richiedere l'accesso al canale i-esimo del DMA e l'inizio di un trasferimento di dati.
- **output DACKx (x = 3-0):** Le *DMA Acknowledge Lines* vengono utilizzate dal controller DMA per permettere ognuna l'accesso al rispettivo canale i-esimo (del DMA) da parte della periferica i-esima e per iniziare un trasferimento di dati.

NOTA: Per le porte che supportano sia input che output, è necessario utilizzare dei particolari accorgimenti, ovvero l'uso di Tri-State Buffers. Tramite i segnali di *output enable* (controllati da CPU o, nel programma realizzato, da TestBench) è possibile stabilire la direzione del trasferimento (input o output), che potrà quindi avvenire solo in un'unica direzione alla volta.

Registri interni

- **reg[3:0] A30_reg**: Registro a 4-bit assegnato alle porte inout A3-A0 (quando l'output è abilitato)
- **reg[7:0] D_reg**: Registro a 8-bit assegnato alle porte inout D7-D0 (quando l'output è abilitato)
- **reg EOP_reg**: Registro a 1-bit assegnato alla porta inout EOP (quando l'output è abilitato)
- **reg[1:0] IOR_MEMR_reg**: Registro a 2-bit assegnato alla porta inout IOR_MEMR_ (quando l'output è abilitato per il DMA)
- **reg[1:0] IOW_MEMW_reg**: Registro a 2-bit assegnato alla porta inout IOR_MEMR_ (quando l'output è abilitato per il DMA)
- **reg HRQ_reg**: Registro a 1-bit assegnato alla porta di output HRQ per permetterne l'assegnazione
- **reg[3:0] A74_reg**: Registro a 4-bit assegnato alle porte inout A7-A4 (quando l'output è abilitato per il DMA)
- **reg ADSTB_reg**: Registro a 1-bit assegnato alla porta di output ADSTB per permetterne l'assegnazione
- **reg AEN_reg**: Registro a 1-bit assegnato alla porta di output AEN per permetterne l'assegnazione
- **reg DACKx_reg (x = 3-0)**: Registri a 1-bit, dove ognuno è associato alla rispettiva porta di output DACK{0,1,2,3}**,** per permetterne l'assegnazione
- **wire[3:0] DREQ_wire**: Usato per raggruppare gli input DREQ3-0, in modo da poterli passare in modo meno verboso al modulo *Priority Encoder*

- **wire[3:0] DACK_wire**: Usato per aggiornare gli output DACK3-0 e per poterli passare in modo meno verboso al modulo *Priority Encoder*
- **wire[1:0] SERVED_CHANNEL**: Usato per prelevare dal modulo del *Priority Encoder* il valore che determina il canale correntemente in uso.
- **reg[7:0] TEMP**: Il seguente registro viene sfruttato per memorizzare temporaneamente il valore dei registri che devono essere bloccati. In particolare:
  - TEMP[3:0] viene utilizzato per memorizzare i valori di A30 una volta che il registro ULLATCH ha valore alto.
  - TEMP[5:4] viene utilizzato per memorizzare i valori di MR[1:0], cioè dei valori usati per codificare il canale che correntemente è in uso
  - TEMP[7:6] viene utilizzato per memorizzare i valori di MR[7:6], cioè dei valori usati per codificare la modalità operativa che usata correntemente
- **reg[7:0] CR**:

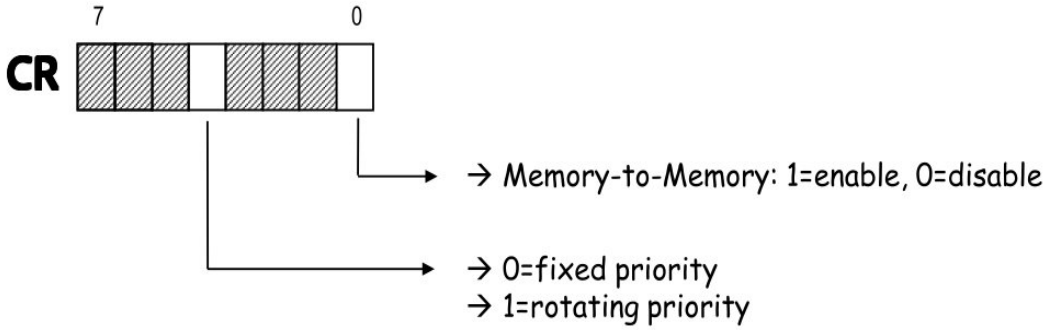

Fig. 2: Configurazione del Command Register (da c124lez16-dma_extended.pdf, slide 33)

- **reg[7:0] SR**:

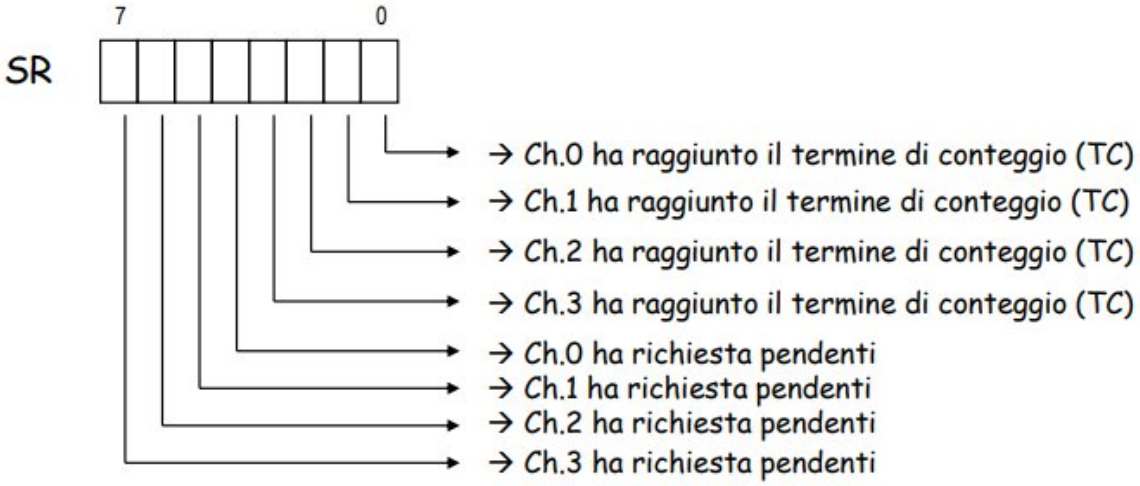

Fig. 3: Configurazione di Status Register (da c124lez16-dma_extended.pdf, slide 34)

- **reg[7:0] MR**:

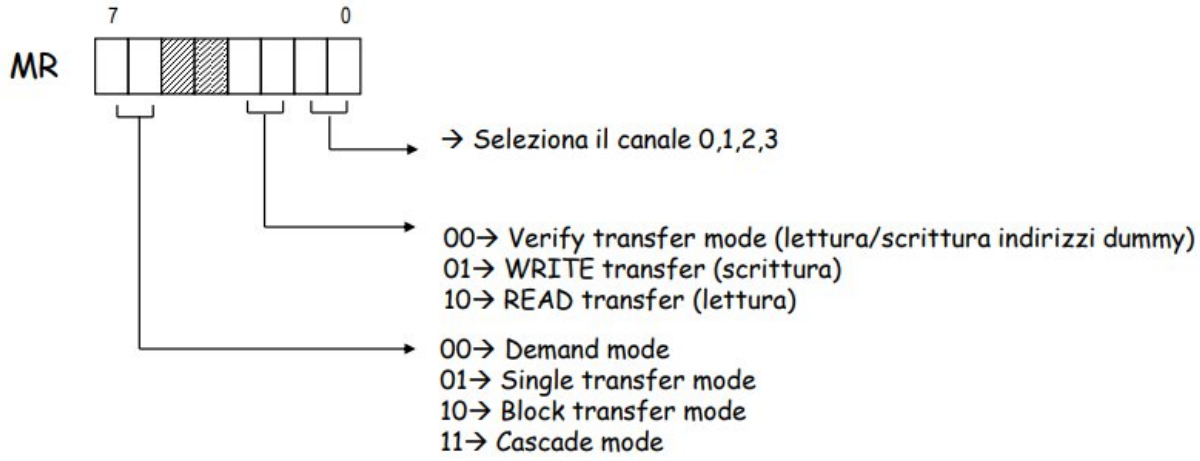

Fig. 4: Configurazione di Mode Register (da c124lez16-dma_extended.pdf, slide 35)

- **reg[7:0] MKR**:

Fig. 5: Configurazione di Mask Register (da c124lez16-dma_extended.pdf, slide 36)

- **reg[15:0] BADDRx(x = 3-0)**: I seguenti registri vengono usati per memorizzare il Base Address associato alla rispettiva porta {0,1,2,3} del DMAC
- **reg[15:0] BWCx (x = 3-0)**: I seguenti registri vengono usati per memorizzare il Base Word Counter associato alla rispettiva porta {0,1,2,3} del DMAC
- **reg[15:0] CADDRx (x = 3-0)**: I seguenti registri vengono usati per memorizzare il Current Address associato alla rispettiva Porta {0,1,2,3} del DMAC. Inizialmente il loro valore è lo stesso del Base Address associato alla stessa porta, successivamente il valore viene incrementato con ogni trasferimento.
- **reg[15:0] CWCx (x = 3-0)**: I seguenti registri vengono usati per memorizzare il Current Word Counter associato alla rispettiva Porta {0,1,2,3} del DMAC. Inizialmente il loro valore è lo stesso del Base Word Counter associato alla stessa porta, successivamente il valore viene decrementato con ogni trasferimento.
- **reg ULLATCH**: Il seguente registro (Upper-Lower LATCH) viene utilizzato per segnalare se la parte trasferita sul bus a 8-bit (D7-D0) è la parte più significativa (ULLATCH = 1'b1) o quella meno significativa (ULLATCH = 1'b0) della word. Viene utilizzato nella trasmissione di BWC{0,1,2,3} e di BADDR{0,1,2,3}.

- **reg CHANNEL_LOCK**: Il seguente registro viene utilizzato per segnalare se è stato selezionato un canale per effettuare il trasferimento. Finché CHANNEL_LOCK assume valore alto, il canale selezionato non può essere modificato a meno che non sia terminato il processo (tramite EOP_) oppure non sia inviato un segnale di reset, quindi gli altri canali risultano inaccessibili fino a quando CHANNEL_LOCK non è disabilitato.
- **reg[2:0] STATE**: I seguenti registri sono utilizzati per memorizzare lo stato di operazione corrente del DMA, seguendo un modello a FSM di Mealy. Gli stati possibili sono i seguenti:
  - SI (3'b111): *Idle State*. In questo stato il DMAC è disponibile per la programmazione e per l'aggiunta in coda di richieste di trasferimento. Si hanno 2 possibili casi in cui il DMAC può essere in questo stato: 1) a seguito di un reset, quando non sono presenti richieste di trasferimento *pending* 2) quando il canale è già stato bloccato: il DMAC continua a poter accettare richieste di trasferimento finché HLDA resta basso.
  - S0 (3'b000): Il DMAC ha ricevuto una richiesta di trasferimento, dunque, richiede il permesso di utilizzare il BUS alla CPU, tramite HRQ. Passa ad SI, che passerà a sua volta a S1 quando HLDA == 1'b1.
  - S1 (3'b001): Una volta ricevuto HLDA alto (passaggio a S1) si verifica che HRQ abbia sempre valore alto. In tal caso si vuole controllare se il trasferimento sia terminato o meno (tramite CWC oppure DREQ, a seconda del caso): se il trasferimento non è terminato valuta se trasmettere su latch esterno il valore di D e passa allo stato S2, altrimenti invia un segnale EOP per segnalare che il trasferimento è terminato. Se HRQ ha valore basso, invece, passa a SI.
  - S2 (3'b010): Transizione a S3, una volta che READY ha valore alto, altrimenti nessuna operazione.
  - S3 (3'b011): Se READY ha valore alto invia (in uscita) l'indirizzo al quale si vuole effettuare la lettura/scrittura, inoltre se ADSTB ha valore alto viene disabilitato (posto a valore basso). Passaggio a S4 una volta che READY ha valore basso.
  - **S4 (**3'b100): In questo stato avviene la lettura dei dati da trasferire (da I/O device a CPU o viceversa), che poi saranno trasmessi al richiedente (a seconda del caso R/W). Inoltre vengono aggiornati i valori per CWC e CADDR. Successivamente, si ha un passaggio incondizionato a S1.
- **reg[7:0] USAGE**: I seguenti registri sono utilizzati per memorizzare la *priority queue* dei canali serviti. Tali registri entrano in uso solo nel caso in cui MR sia programmato in modo da avere *Rotating Priority*.

Reset

Tutti i registri interni sono impostati al valore di default (ovvero con tutti i bit nulli o indeterminati). Inoltre viene impostato EOP_ = 1'b0, in modo da poter evitare l'implementazione di reti logiche superflue: è possibile sfruttare direttamente il funzionamento del caso di *End of Process*. Lo svantaggio di tale ottimizzazione è che l'operazione richiede un ulteriore ciclo di clock per completare il reset.

End of Process

Imposta il canale correntemente in uso in base al valore fornito da parte del *Priority Encoder*, inoltre, se si hanno delle richieste *pending* viene garantito che venga servita la prossima richiesta. Se invece nessuna richiesta è *pending*, imposta lo stato a SI. Infine, nel caso in cui la *Rotating Priority* sia abilitata aggiorna i registri USAGE, in modo che il *Priority Encoder* possa calcolare il nuovo canale che deve essere servito. Terminate queste operazioni il registro EOP_reg viene abilitato (EOP_reg = 1'b1), per segnalare che l'*End Of Process* è stato servito.

Programmazione iniziale (tramite A30)

In base ai valori di A3-A0 si ha:

```
A[3:0] = 0000    R/W   (BA0)   W: Scrittura dell'indirizzo base (B)
                                R: Lettura dell'indirizzo attuale (current Address)
A[3:0] = 0001    R/W   (WC0)   W: Scrittura dei byte da trasferire
                                R: Lettura dei byte che restano da trasferire
A[3:0] = 0010    R/W   (BA1)   W: Scrittura dell'indirizzo base (B)
                                R: Lettura dell'indirizzo attuale (current Address)
A[3:0] = 0011    R/W   (WC1)   W: Scrittura dei byte da trasferire
                                R: Lettura dei byte che restano da trasferire
A[3:0] = 0100    R/W   (BA2)   W: Scrittura dell'indirizzo base (B)
                                R: Lettura dell'indirizzo attuale (current Address)
A[3:0] = 0101    R/W   (WC2)   W: Scrittura dei byte da trasferire
                                R: Lettura dei byte che restano da trasferire
A[3:0] = 0110    R/W   (BA3)   W: Scrittura dell'indirizzo base (B)
                                R: Lettura dell'indirizzo attuale (current Address)
A[3:0] = 0111    R/W   (WC3)   W: Scrittura dei byte da trasferire
                                R: Lettura dei byte che restano da trasferire
A[3:0] = 1000    W     (CR)    --> Command Register
A[3:0] = 1000    R     (SR)    --> Status Registe
A[3:0] = 1100    W     (ULLATCH)  Clear-Byte-Pointer-FlipFlop (set ULLATCH to 1'b0)
A[3:0] = 1011    W     (MR)    --> Mode Register
A[3:0] = 1111    W     (MKR)   --> Mask Register
Nei registri BA_i e WC_i viene scritto prima LSB e poi MSB, inoltre all'inizio deve
essere dato un comando di "Clear-Byte-Pointer-FlipFlop", ovvero di scrittura
all'indirizzo 1100.
```

Per la programmazione dei canali 3-0, avendo selezionato l'operazione da effettuare tramite la linea A30 (scrittura sul canale i-esimo del Base Address o del Base Word Counter oppure lettura sul canale i-esimo del Current Address o del Current Word Counter), anzitutto è necessario comprendere se si vuole effettuare un'operazione di lettura o scrittura, tramite il valore (in ingresso) della porta IOW_MEMW_[1:0].
Successivamente viene sfruttato il registro interno ULLATCH per determinare se i dati che sono trasferiti sono la parte alta della half-word (16-bit) oppure la parte bassa. Nel caso in cui il trasferimento coinvolga la parte bassa dei dati che si vogliono trasmettere, allora verrà sfruttato il registro TEMP[3:0] per memorizzare i valori di A30 che codificano l'operazione appena iniziata, in modo da poter bloccare il valore di A30 al ciclo successivo: facendo così è possibile svolgere il trasferimento in 2 fasi (una per ciclo di clock), dove in ogni fase si vogliono trasferire 8 bit tramite la linea dati D (nella prima fase i bit bassi, nella seconda fase i bit alti).
Nel caso in cui, invece, venga trasferita la parte alta (ULLATCH == 1'b1 --> (dopo il trasferimento diventa:) ULLATCH == 1'b0), allora a seguito della lettura dei dati inviati viene aggiornato lo Status Register per indicare che si ha una richiesta *pending*. Dunque vengono aggiornati i registri USAGE, che tengono conto della coda delle richieste di trasferimento secondo la *Rotating Priority*.

Infine il DMAC entra nello stato S0. In particolare, l'ingresso nello stato S0 avverrà soltanto una volta che viene inviato l'indirizzo a cui effettuare il trasferimento, dunque sarà necessario impostare in anticipo il valore del Base Word Counter.

Nel caso in cui, invece, si vogliano trasferire i dati per programmare uno dei registri interni, usati per controllare il DMAC, il trasferimento richiederà un unico ciclo di clock, dato che i registri sono a 8-bit, dunque sarà sufficiente un unico trasferimento a 8-bit sulla linea di dati.

Active Cycle del DMAC

Modalità *Demand*

Nella modalità *Demand* il DMAC si comporta analogamente alla modalità *Block Transfer*, tuttavia non sarà necessario l'uso dei registri *Word Counter*, dato che il trasferimento dovrà proseguire finché non sarà inviato esternamente un segnale di *End of Process* oppure finché non verrà disabilitato, da parte della CPU o della periferica, il segnale DREQ per la porta in uso. In questo tipo di trasferimento serve fare particolare attenzione alla durata nel processo, dato che nel caso in cui non venisse arrestato, lo scambio di dati potrebbe continuare anche dopo aver raggiunto la fine della memoria del dispositivo connesso.

Modalità *Single Transfer*

Nella modalità *Single Transfer* è necessario fornire al DMA Controller l'indirizzo per eseguire un unico trasferimento di dati alla volta, ripetendo il processo finché non sarà stato trasferito il numero di *word* richiesto (tale numero è memorizzato nel *Current Word Counter*). Una volta che sono stati programmati i registri che contengono l'indirizzo del trasferimento ed il numero di parole da trasferire, si ha la garanzia che dopo ogni singolo trasferimento il DMAC ritornerà nello stato SI, mantenendo il blocco del canale, per attendere un nuovo indirizzo per il prossimo trasferimento. Dunque l'unico controllo per determinare se terminare il processo sarà quello di verificare se il *Current Word Counter* sia arrivato a "0": in tal caso si avrà l'*End of Process*.

Il vantaggio di questa modalità è che permette di svolgere il trasferimento di più dati che provengono da indirizzi anche non adiacenti, dato che è possibile selezionare esattamente l'indirizzo per ogni singolo trasferimento.

Modalità *Block Transfer*

La modalità *Block Transfer* consiste nel programmare il DMA Controller per eseguire un trasferimento "a blocchi", ovvero di una o più singole word (8-bit), leggendo, ripetutamente, word da 8-bit ad indirizzi successivi, senza necessità di inviare un nuovo indirizzo ogni volta. Inoltre, è importante osservare che gli 8 bit legati alla parte alta dell'indirizzo da inviare saranno salvati sul latch a 8-bit solo nel caso in cui essi subiscano una variazione (durante la trasmissione del primo indirizzo oppure ogni volta che i bit bassi del canale i-esimo hanno raggiunto il valore massimo, cioè CADDRi[7:0] == 8'hFF).
Difatti i registri *Current Word Counter* e *Current Address* saranno aggiornati dal DMA alla fine di ogni operazione di trasferimento di singola word. Nel momento in cui il *Current Word Counter* viene azzerato (decrementando di "1" con ogni word trasferita), viene inviato un *End of Process*. Si osservi che anche in questo caso il canale corrente viene bloccato tramite il registro CHANNEL_LOCK, finché non viene inviato EOP_ = 1'b0.

Modalità *Cascade*

La modalità *Cascade* permette di sfruttare la connessione in cascata di più DMAC, in modo da poter estendere il numero di canali a disposizione. Le porte HQR e HLDA del dispositivo successivo sono collegate al DREQ e DACK di uno dei canali del primo DMAC. Il canale del DMAC a monte, che viene utilizzato per il collegamento, non

dovrà generare nessun tipo di indirizzo o di segnale di controllo, inoltre le sue linee DREQ e DACK si occuperanno di gestire le varie richieste del DMA provenienti dalle linee HRQ e HLDA del secondo DMAC a cui sono connesse.

NOTA: A prescindere dalla modalità di trasferimento selezionata, una volta che il trasferimento è terminato viene impostato HRQ = 1'b0: finché HLDA non sarà disattivato non sarà possibile iniziare un nuovo trasferimento (su un canale qualsiasi).

Modulo *Priority Encoder*

Il modulo *Priority Encoder* è la componente dell'Intel 8237A che si occupa di gestire le priorità assegnate alle richieste pendenti dei vari canali, le quali necessitano di avere un ordine assegnato per essere servite da parte del DMAC. Il *Priority Encoder* potrà operare secondo due diverse modalità:

- **Fixed Priority**: Il Canale 0 ha la massima priorità, dunque, se è presente una richiesta su di esso, sarà sempre il primo canale ad essere servito. Seguendo l'ordine (decrescente) del livello di priorità, il Canale 1 è il successivo, che avrà priorità più alta del Canale 2, che a sua volta risulta avere maggiore priorità rispetto al Canale 3. (Analogo ad un *Priority Encoder* (lez 3, sl. 7), con abilitazione basata sui segnali DREQ).
- **Rotating Priority**: Viene valutata, per ogni canale, la rispettiva priorità tramite il registro USAGE. USAGE[1:0] descriverà la priorità associata al Canale 0 (con un valore intero tra 0 e 3, memorizzato su 2 bit, dove 0 sarà la massima priorità e 3 la minima priorità), USAGE[3:2] quella del Canale 1, USAGE[5:4] quella del Canale 2 ed infine USAGE[7:6] quella del Canale 3. Nel caso di un iniziale pareggio a livello di priorità (dopo il Reset, infatti, tutti i canali avranno uguale priorità) verrà seguito lo stesso schema implementato per la *Fixed Priority*.

Nella modalità *Rotating Priority* il DMAC garantisce la *fairness*, ovvero che tutte le richieste non debbano attendere più di 3 turni prima di essere servite.

TestBench

Il TestBench è fondamentale per verificare che il modulo realizzato operi nel modo corretto. Nel caso del seguente progetto, il TestBench deve ricoprire sia il ruolo della CPU, la quale si occupa della programmazione iniziale del DMAC, in modo che successivamente possa lasciare che esso operi autonomamente, senza la necessità di impiegare le risorse della CPU per il trasferimento, sia il ruolo della periferica (o dispositivo di I/O) connessa al PC. Il secondo ruolo del TestBench, in particolare, risulta utile per verificare che i valori delle word e degli indirizzi inviati dal Controller siano corretti.

Inoltre, anche la gestione degli *Output Enable* dei Tri-State Buffer sarà compito del TestBench.

Sono stati realizzati i TestBench per le seguenti operazioni:

- Single Transfer, Write Mode, Fixed Priority, Channel 0 and Channel 1, Not Memory-To-Memory (1 word on Channel 0, 2 words on Channel 1). (si veda: "single_transfer_fixed_priority.png")
- Single Transfer, Write Mode, Rotating Priority, Channel 0 and Channel 1, Not Memory-To-Memory (1 word on Channel 0, 2 words on Channel 1). (si veda: "single_transfer_rotating_priority.png")
- Block Transfer, Read Mode, Fixed Priority, Channel 0, Not Memory-To-Memory (3 words) (si veda: "block_transfer.png")
- Demand Mode, Read Mode, Fixed Priority, Channel 0, Not Memory-To-Memory (3 words, poi viene inviato DREQ0 = 1'b0 per terminare il trasferimento) (si veda: "demand_mode.png")

NOTA: Le immagini dei diagrammi di temporizzazione sono state allegate in una cartella apposita.