Figura X – Schema dell’entity CMD\_CTRL

Il controllore, chiamato CMD\_CTRL, è il centro di controllo dell’intera scheda di sviluppo. Si occupa di leggere il comando dalla FIFO\_RX, interpretarlo, eseguirlo e inviare sull’UART una risposta, detta report.

La macchina presenta una forma a FSMs annidate per cui la FSM del controllore gestisce più macchine che in vari stadi lavorano in parallelo per riconoscere ed eseguire i comandi.  
Il funzionamento del controllore può essere riassunto in tre fasi principali: fetch/decode – execute – report.  
Le sottomacchine possono divise in due gruppi FD, abbrev. di Fetch/decode, e EX, abbrev. di Execute. Ogni macchina del primo gruppo è accoppiata con una del secondo gruppo e se una delle macchine del gruppo FD va a buon fine, viene attiva la sola macchina corrispondente nel gruppo EX.

Figura X^2 – Diagramma di stato della FSM di CMD\_CTRL

Nella prima fase, il controllore si occupa solo di fornire i caratteri al gruppo FD. Queste macchine si occupano di riconoscere il comando e di estrapolare i campi con i dati. In un normale funzionamento, ogni macchina legge il comando ed una sola lo riconosce attivando il proprio segnale di MATCH.  
Il controllore quando riconosce un terminatore di linea, cioè i caratteri LF e RT, che fanno attivare il segnale EOL, entra in uno stato di attesa. Da qui vi può uscire se arriva un carattere non EOL per cui è iniziato un altro comando o quando scatta un time-out che dura quanto la trasmissione di due caratteri.  
Il controllore poi analizza la situazione:  
se una sola macchina ha dato un MATCH, entra nella fase di execute ed attiva la sola macchina corrispondente;  
se nessuna macchina ha dato un MATCH significa che il comando è sconosciuto e il controllore entra nel suo ciclo di gestione;  
se più di una macchina da un MATCH o almeno una macchina segnala un errore, si entra nel ciclo di errore.

Nel caso di esecuzione corretta, la macchina verrà avvertita con un segnale DONE\_EX della fine della fase. Se non si verificano errori, per cui si entra nel ciclo di gestione, si va nella fase di report. Qui per prima cosa viene resettata l’entity che scrive sulla FIFO\_TX, cioè la macchina CHAR\_WRITER \_FIFO. Poi si seleziona su un multiplexer l’unico messaggio da inviare, di conseguenza tutti i tipi possibili di messaggio di responso vengono sempre generati ma solo quello corretto viene instradato per essere scritto. Inoltre si dà all’entity che scrive anche il numero di caratteri proprio per ogni messaggio. Infine si attiva CHAR\_WRITER\_FIFO, che quando ha terminato attiva il segnale DONE\_CHWR e il controllore torna alla fase di fetch/decode. Nel caso di errore o comando sconosciuto vengono generati due segnali, ERR e UNK rispettivamente, e viene selezionato il messaggio corrispondente.

Le macchine del gruppo FD sono tutte dei riconoscitori di sequenze e presentano una struttura simile. Sono raggruppate in modo che riconoscano comandi che hanno funzioni simili tra loro. Tutte hanno in comune i segnali MATCH, DONE ed ERR\_FD. Il primo si attiva se viene riconosciuto il comando corrispondente. Il secondo si attiva se è avvenuto un match o il messaggio è sconosciuto e di conseguenza comunica che questa macchina non ha più necessità di altri caratteri. Infine il terzo avverte di un errore.  
Inoltre queste macchine hanno in comune i controlli per l’entity di lettura CHAR\_FEEDER\_FIFO, RDY e GOT. A livello del controllore questi segnali vengono combinati così che tutte le entity possano leggere i caratteri correttamente.

Figura X^3 – Diagramma di stato di FIELD\_RECOGNIZER

Al centro di queste macchine vi è l’entity FIELD\_RECOGNIZER. Quest’entity riconosce il numero di campi e scandisce l’ordine delle altre FSM collegate. Anche in questo caso vi è una struttura a FSMs annidate in cui ci sono tanti gruppi di macchine quanti il numero massimo di campi dei comandi riconoscibili.

Ogni istruzione va da un minimo di un campo a tre campi. La macchina li riconosce perché sono separati da caratteri non stampabili, come lo spazio o la tabulazione che sono individuati dalla disattivazione del segnale PR, abbrev. di Printable.  
La macchina inizialmente viene resettata e poi entra nel ciclo di lettura dei caratteri, OK. Appena arriva un carattere non stampabile entra in un ciclo di attesa, WOKNP e OKNP, che termina appena arriva un carattere stampabile. Poi incrementa un contatore che fornisce un segnale one-hot di attivazione ai gruppi di macchine, INC. Il contatore fornisce anche in uscita il numero dei campi letti. Un solo gruppo di macchine è attivo alla volta e quando sono disattivate rimangono immobili senza modificare lo stato delle uscite.  
Se all’arrivo del carattere non stampabile, l’indice del campo del comando rientra in un intervallo prefissato vi è il match del numero di campi e si entra nel ciclo che gestisce questo caso. Il ciclo funziona allo stesso modo del precedente solo con l’aggiunta di uno stato di controllo in più, MTCCHK, che controlla se al prossimo incremento si supera l’intervallo di campi consentiti. Di conseguenza, il comando è sconosciuto e si entra nel proprio ciclo di gestione e la macchina vi rimane intrappolata fino al prossimo riavvio.

I gruppi di macchine controllate da FIELD\_RECOGNIZER sono formati da più istanze, con piccole variazioni, delle tre stesse macchine generiche. La prima riconosce un messaggio fisso salvato in una ROM e si chiama STATIC\_RECOGNIZER. La seconda riconosce un campo di lunghezza variabile che può contenere valori diversi, come ad esempio l’indirizzo a cui leggere, e si chiama DYNAMIC\_RECOGNIZER. Quest’ultima salva i caratteri del campo in uno shift register. La terza, che si chiama ONECHAR\_RECOGNIZER, è una variazione della seconda e differisce solo perché il campo ha lunghezza unitaria e controlla un registro invece che uno shift register.

STATIC\_RECOGNIZER

Figura X^4 – Schema dell’entity STATIC\_RECOGNIZER assieme ad una ROM

STATIC\_RECOGNIZER è un classico riconoscitore di sequenze reso universale dal generic LEN assieme al collegamento ad una ROM dedicata che contiene i caratteri del messaggio da riconoscere.  
Per prima cosa la macchina dev’essere riavviata e a questo punto, stato OK, ad ogni nuovo carattere lo confronta con quello presente in memoria e nel caso siano uguali attiva il segnale COR. Se il carattere è sconosciuto va in un ciclo di attesa sino al prossimo riavvio, RDUNK e UNK che sono separati per generare i segnali RDY e GOT. Altrimenti la macchina va in uno stato per generare l’impulso di GOT ed incrementare il contatore che contiene l’indirizzo della ROM. Se si arriva sino alla fine del messaggio, si passa agli stati WTMTC, per generare il segnale GOT, e poi a MTC che attiva il segnale MATCH. Se arriva un altro carattere, il messaggio è sconosciuto.

Figura X^5 – Diagramma di stato dell’entity STATIC\_RECOGNIZER

DYNAMIC\_RECOGNIZER

Figura X^6 – Schema dell’entity DYNAMIC\_RECOGNIZER

A differenza di STATIC\_RECOGNIZER, quest’entity deve riconoscere e salvare un campo di lunghezza e contenuto variabili. Inoltre, per rendere la macchina riutilizzabile, non genera da sola il segnale COR ma gli viene fornito dall’esterno. Per esempio, molti campi possono di numeri esadecimali, per cui la logica che genera COR controlla se il carattere in ingresso è una cifra esadecimale attivando il segnale o meno.

La macchina per funzionare dev’essere prima resettata, poi entra nel ciclo per i caratteri corretti, OK. Se arriva un carattere e COR è attivato, lo carica nello shift register attivando il segnale SHEN, abbrev. di Shift enable. Se COR è disattivato invece va nel ciclo per i comandi sconosciuti. Nel caso in cui la lunghezza del campo sia in un range definito con i generics, entra nel ciclo di match che attiva il segnale MATCH.

Figura X^7 – Diagramma di stato dell’entity DYNAMIC\_RECOGNIZER

Comandi

Il controllore riconosce 5 diversi gruppi di comandi: Write, Read, Clock, Report e Reset.  
Il controllore per comunicare che è pronto a ricevere un comando, invia sull’UART l’intestazione “> “, che è per l’appunto la prima operazione svolta all’inizio del ciclo della FSM. Dopodiché esegue un eco di battitura del comando che riesce a leggere.

Durante la fase di report, mette come intestazione un carattere speciale e fa seguire a tutto il messaggio inviato un’intera riga vuota. La presenza del carattere speciale servirebbe per una futura automatizzazione delle operazioni con il controllore poiché possono essere usati come tag per riconoscere l’operazione svolta.

Per tutti i caratteri, il controllore è case insensitive.

Gruppo Write

Il gruppo Write si occupa di scrivere i dati passati sulle memorie collegate allo Z80X e fa uso del carattere speciale “@” che significa asserzione al comando inviato.  
Sono disponibili due modalità, una scrive una sola riga, l’altra permette di usare un comando più corto, quindi più veloce, e scrive il dato di seguito al precedente. L’ultima modalità è utile per scrivere molte righe di seguito e fa uso di un contatore per selezionare l’indirizzo.

Per scrivere una singola riga si fa uso del comando:

WR <dd> <aaaa>  
<dd> : è il dato da scrivere espresso come esadecimale. Può essere lungo 1 o 2 caratteri.  
<aaaa> : è l’indirizzo del a cui scrivere espresso come esadecimale. Può essere lungo fino a 4 caratteri.

La macchina carica il valore aaaa nel contatore e scrive il dato dd all’indirizzo puntato.

Per la scrittura multipla, vi sono due istruzioni: la prima setta il contatore mentre la seconda esegue la scrittura all’indirizzo puntato dallo stesso.

WRL <aaaa> Imposta il valore del contatore.

W <dd> Scrive il valore dd all’indirizzo puntato da dal contatore.

In ogni caso dopo una qualsiasi operazione di scrittura, il contatore viene incrementato e la macchina risponde con “@”.

Gruppo Read

Il gruppo si occupa di stampare sull’UART il valore delle righe selezionate della memoria. Prima dell’opportuna risposta, inserisce il carattere speciale “#” che identifica il gruppo.

Ci sono tre comandi:

RD <aaaa> Legge la sola riga puntata da aaaa.

RDL <aaaa\_1> <aaaa\_2> Legge l’intervallo di righe tra aaaa\_1 e aaaa\_2.

RDM ALL | ROM | ROM Legge l’intervallo corrispondente alla sezione selezionata.

Al primo comando la macchina legge la riga corrispondente e scrive sull’UART il dato come numero a due cifre esadecimale.  
Negli altri due comandi invece scrive una o più righe secondo lo schema <line\_of\_data>:

<line\_of\_data> : <dd> {<dd> …}

Le <line\_of\_data> sono linee che contengono un numero prefissato di dati settato con il generic DATAxLINE. Per cui fintato che ci sono dati da inviare e il numero su quella linea è inferiore a DATAxLINE, vengono scritti i dati come coppie di cifre esadecimali seguite da due spazzi. Se il dato è l’ultimo della linea, viene seguito dalla coppia di caratteri terminatori di linea.

Nel caso di RDM, con i tag vengono selezionati degli intervalli prefissati. Questi sono generati dal numero di bit dell’indirizzo della memoria che è passato all’entity come il generic N. Per cui si ha:

ALL : aaaa\_1 = 0, aaaa\_2 = 2^N – 1  
ROM : aaaa\_1 = 0, aaaa\_2 = 2^(N-1) -1  
RAM : aaaa\_1 = 2^(N-1), aaaa\_2 = 2^N -1

La macchina che ne esegue la decodifica dell’operazione può generare errore se aaaa\_2 è minore di aaaa\_1 o almeno uno dei due esce dalle dimensioni della memoria, cioè fuori dall’intervallo [0, 2^N-1].

Gruppo Clock

Sono le istruzioni che controllano l’entity CLK\_GEN e quindi il clock dello Z80X. Ne fanno parte quattro istruzioni.

CLK START | STOP Che attiva o disattiva l’uscita del clock verso lo Z80X

CLKDIV <n> Che setta il divisore di frequenza con cui si ottiene il clock d’uscita  
<n> : è un valore esadecimale da 0 a 2^DIV\_N -1. DIV\_N è un generic. Il numero massimo di cifre si ottiene con la parte alta della divisione di DIV\_N per 4.

CLKST <s> Che setta il numero di cicli di clock d’uscita per cui far andare il segnale prima di fermarsi.  
<s> : è un valore esadecimale da 0 a 2^STEP\_N -1. STEP\_N è un generic. Il numero massimo di cifre si ottiene con la parte alta della divisione di STEP\_N per 4.

M1 <s> Che setta il numero di fronti negativi di nM1 per cui far andare il segnale di clock prima di fermarsi.

A tutti i comandi viene risposto con il carattere di asserzione “@”.

La macchina di fetch/decode può generare errore se il valore di n o s è oltre il valore massimo. Questo perché con il numero di cifre a disposizione si possono rappresentare numeri più grandi di quelli consentiti.

Gruppo Reporter

Il gruppo reporter si occupa di gestire il report dello stato dello Z80X. Vi fanno parte tre istruzioni:

REPEN Che abilita la sezione di snap dello Z80X

REPSEL <hh> Che abilita o meno i trigger sul cambiamento di alcuni campi dello snap  
<hh> : è una coppia di cifre esadecimali che rappresentano un numero binario a 7 cifre. Ogni cifra corrisponde al segnale di abilitazione di un campo specifico.

REP Che inizia l’invio di tutti gli snap acquisiti sino all’invio di questo comando.

Ai primi due comandi, la macchina risponde con il carattere di asserzione “@”.  
Al comando REP invece segue l’invio di tutte le righe di snap presenti. Le righe di snap sono formate nel seguente modo:

$ <CNT\_MSTR> <CNT\_CLK> <CTRL\_BUS> <A> <DIN> <DOUT>

Il carattere “$” segnala l’inizio di una riga di snap. Segue un campo di 4 cifre esadecimali che rappresentano il conteggio su 16 bit dei cicli di clock master, a 4MHz, contati dall’inizio. Questo permette di avere un riferimento temporale dei singoli snap. Il campo seguente è un altro riferimento temporale rispetto al clock dello Z80X, occupa una sola cifra ed è su 3 bit.  
Segue una parola di 4 caratteri che rappresenta la giustapposizione di tutti i segnali di controllo dello Z80X , sia in ingresso che in uscita, rappresentati su 13 bit.  
Come ultimi campi vi sono i valori dei bus A, DIN e DOUT rispettivamente su 4 e 2 cifre.  
Ogni campo è separato da uno spazio e la riga termina con i due caratteri terminatori.

Con l’istruzione REPSEL si selezionano quali campi attivino il trigger di Z80\_SNAPPER per creare un altro snap. I gruppi di segnali che variando possono generare un segnale di trigger sono: nRESET, System Control, CPU Control meno nRESET, CPU Bus Control, A, DIN e DOUT. La macchina di fetch/decode genera errore se il valore di hh non è rappresentabile su 7 bit.

Gruppo Reset

In questo gruppo vi è l’unica istruzione RESET che attiva il ciclo di reset. Il ciclo inizia impostando gli step del segnale CLK di CLK\_GEN su 5 cicli, così da essere sicuri che lo Z80X si resetti, e si mantiene attivo il segnale nRESET fintantoché il clock sta andando.

Errore e Comando sconosciuto

In caso avvenga un errore o il comando sia sconosciuto il controllore risponde sull’UART con due comandi dedicati.

Per l’errore risponde con:

&Err <ERR\_CODE> Che con il carattere speciale “&” segnala un errore e con ERR\_CODE informa chi l’ha generato

<ERR\_CODE> : è un numero binario di 7 cifre in cui i bit in ordine dal meno significativo segnalano :  
ERR\_RD\_FD, errore da parte del fetch/decode del gruppo Read;  
ERR\_RD\_EX, errore da parte dell’execute del gruppo Read, per cui la memoria non risponde alla lettura;  
ERR\_WR\_FD, errore da parte del fetch/decode del gruppo Write;  
ERR\_WR\_EX, errore da parte dell’execute del gruppo Write, per cui la memoria non risponde alla scrittura;  
ERR\_CLK\_FD, errore da parte del fetch/decode del gruppo Clock;  
ERR\_REP\_FD, errore da parte del fetch/decode del gruppo Report;  
ERR\_REP\_EX, errore da parte dell’execute del gruppo Report, per cui la FIFO non risponde alla lettura;

In caso di comando sconosciuto il controllore risponde con:

!Unk cmd

In cui il carattere “!” segnala lo stato.