Per rappresentare i diagrammi di stato userò la seguente convenzione:  
gli stati sono rappresentati per mezzo di ellissi e le transizioni con frecce su cui è riportata la condizione;  
sotto agli stati, solitamente in verde, sono riportate le uscite che vengono attivate in quello stato. In arancio invece sono evidenziate quelle uscite attivate che sono utili alla struttura annidata delle macchine. In fucsia le uscite che hanno un comportamento particolare;  
alcuni insieme di stati sono raggruppati in rettangoli dai bordi smussati che presentano lo stesso comportamento di un normale stato;  
per non appesantire troppo di transizioni il diagramma, alcuni salti a stati o gruppi vengono indicati con una freccia verso il nome della destinazione generalmente azzurro;  
in diagrammi in cui è importante la relazione con i cicli di clock, questi sono evidenziati da sezioni viola che riportano il nome del T-cycle corrispondente e lo stato di CLK.

Il ciclo Main dello Z80X è gestito da due FSM: Master e Main.

Master

Figura X – BUS request/Acknowledge Cycle [Z80.pdf p.25]

Master gestisce il comportamento del microprocessore quando si verifica una richiesta del bus e presenta solamente due stati: RUNNING e BUSREQUEST. La macchina è temporizzata solo sul fronte positivo di CLK, poiché le transizioni che deve fare avvengono solo su quell’evento a differenza di Main e delle altre che sono temporizzate su entrambi i fronti.  
Durante il normale funzionamento di Z80X, Master si trova nello stato RUNNING. Come si vede dalla Figura X, dopo il ciclo di clock finale di un qualsiasi ciclo macchina, quindi quando si attiva MLAST\_NEDGE, se l’ingresso nBUSREQ è attivo Master passa allo stato BUSREQUEST. In questo stato blocca il funzionamento della FSM Main, disattiva forzatamente i segnali nM1 e nRFSH e attiva tutti i segnali che mettono i bus in alta impedenza. Contemporaneamente segnala alla periferica richiedente che la richiesta è stata servita attivando nBUSACK.  
La macchina esce dallo stato BUSREQUEST solamente nel se nBUSREQ è disattivato tornando allo stato RUNNING e facendo ripartire la FSM Main.  
Questa configurazione permette di fermare il ciclo di funzionamento di Z80X ripartendo dallo stesso stato senza avere troppi stati aggiuntivi e permette di dare alle richieste la priorità massima.

Main

Figura XX – Diagramma di stato della FSM Main con raggruppati gli stati affini.

La FSM Main è la macchina che supervisiona il comportamento dello Z80X e può cambiare di stato solamente quando Master è in RUNNING.  
La macchina controlla altre sottomacchina, dette anche uFSM abbrev. di Micro FSM, attraverso due segnali TRIG e DONE seguiti dal suffisso della macchina a cui fanno riferimento.

Main presenta un ciclo ordinario che può essere riassunto in tre sezioni principali: fetch – decode – execute.

Figura Y – Diagrammi di stato dei cicli FETCH e FETCH EXTIR.

Le fasi di fetch – decode possono essere singole o estese e sono raggruppate in due gruppi. Entrambi i gruppi presentano la stessa struttura: prima uno stato, FET, che avvia la uFSM per il fetch, chiamata OPFET, poi uno stato, FETWT, che attende la conclusione del ciclo di fetch e l’inizio del refresh seguito dallo stato di decodifica, DEC. Però i due gruppi differiscono poiché nel secondo caso, IR non viene resettato prima della lettura del dato così da ottenere l’istruzione estesa. Per questo gli stati del secondo gruppo presentano l’interfisso -EXT- che ne segnala la differenza.  
Negli stati DEC e DECEXT, che eseguono la fase di decode, si attende che la macchina del fetch concluda completamente il suo ciclo e vengono incrementati sia PC che R ma non vengono svolte operazioni sull’esterno. Nel frattempo viene abilitato il DECODER che leggendo l’IR genera il gruppo FLOWCTRL. Fanno parte di questo gruppo cinque segnali chiamati EXiEN, abbrev. di Execution i Enable, in cui la i è sostituita dall’indice della fase di esecuzione corrispondente, assieme ai segnali IOnMEM, RDnWR, DONE\_EX.  
Può avvenire che un’istruzione sia istantanea cioè che l’operazione richiesta sia svolta direttamente nella fase di decode che quindi non coinvolge nessuna operazione sull’esterno. Un esempio sono le istruzioni di caricamento di un registro in un altro come LD A, B oppure le operazioni aritmetiche a 8 bit come ADD A, B. In questo caso, la macchina non entra nella fase di execute e torna direttamente al fetch.

La fase di esecuzione è formata da 5 blocchi elementari che sono identici a meno dell’indice che ne determina l’ordine. Sono cinque poiché al massimo lo Z80, e lo Z80X di conseguenza, esegue al massimo cinque operazioni R/W e di pura esecuzione differenti.

Figura YY – Diagrammi di stato dei cicli EXnEN, n indica l’indice della fase di esecuzione corrispondente.

Nei blocchi elementari ci sono due percorsi paralleli: uno gestisce le sottomacchina per la lettura o scrittura, mentre l’altro è uno stato di attesa di un’esecuzione generica, svolta internamente alla CPU senza coinvolgere l’esterno. Il primo è formato da due stati RDWRi e RDWRWTi, con i che indica sempre l’indice della fase. Nel primo stato avviene l’avvio della sottomacchina per le operazioni della memoria, MEMRDWR, se il segnale IOnMEM di FLOWCTRL è disattivato al contrario viene avviata la macchina per le operazioni sulle periferiche, IORDWR. Lo stato successivo attende il completarsi del ciclo ed esegue i controlli sui FFs degli interrupt e su EX(i+1)EN così da indirizzare lo stato verso il proseguimento corretto dettato da DECODER.  
Lo stato di esecuzione EXi, con i sempre indice della fase, è regolato da un contatore che si incrementa ad ogni evento di CLK scandendo le azioni all’interno dello stato. Il DECODER attiva il segnale DONE\_EX al raggiungimento di valori specifici che corrispondono alla loro effettiva durata, definita dall’istruzione che si sta svolgendo, e all’uscita dallo stato EXi.  
La selezione di un percorso od un altro è determinata dal valore di EXiEN che è un vettore di due bit in cui il più significativo abilita il percorso di R/W mentre l’altro il percorso di esecuzione interna. Se il vettore è pari a 00, quindi nessuna delle due è stata abilitata, significa che l’istruzione è arrivata al termine e si può procedere con la successiva fase di fetch, richiudendo il ciclo principale.  
Il caso dell’istruzione istantanea avviene con EX1EN = 00 per cui non è abilitato nessun percorso nella fase di esecuzione successiva.  
Dall’ultima fase di esecuzione, EXEC5, non è presente nessun’altra fase successiva per cui la macchina torna alla fase di fetch.

Figura X^3 – Halt Acknowledge [Z80.pdf p.26]

Figura YYY – Diagrammi di stato del ciclo HALT CYCLE.

Un’istruzione particolare è HALT che vale 0076H. Quest’istruzione viene eseguita direttamente dallo stato DEC che la riconosce e mette la macchina nel ciclo di sosta, HALT CYCLE. In questo ciclo ci sono quattro stati, uno per ogni T-cycle, che fanno eseguire allo Z80X delle operazioni NOP, abbrev. di No Operation, in cui avviene solamente il refresh della RAM. Quando la CPU entra in questo ciclo attiva il segnale nHALT per comunicare che è in attesa dall’esterno. Lo Z80X può uscire da questo stato solo per mezzo di un NMI o di un INT se abilitato oppure con nRESET. La lettura dei corrispondenti FFs viene fatta solo nel quarto ed ultimo stato del ciclo.

Figura X^4 – Power-Down Acknowledge [Z80.pdf p.31]

La presenza del ciclo HALT è utile per mettere in stand-by la CPU mantenendo comunque la RAM aggiornata.  
Si può inoltre mettere la CPU in uno stato di power-down così da farle consumare meno corrente. Per far ciò, alla fine di un ciclo di HALT si deve togliere il segnale di CLK. In questo modo la CPU consuma il valore minimo di corrente ma non aggiorna la RAM. Per uscire da questo stato si fa allo stesso modo del precedente fornendo però prima il segnale di CLK.

Durante l’esecuzione di un INT in Mode 0, le operazioni di decode ed execute vengono svolte allo stesso modo di quelle normali.

Figura Y^4 – Diagrammi di stato dei cicli NMI CYCLE e INT ACK.

Nel caso ci si trovi in uno stato terminale cioè DEC e DECEXT con EX1EN = 00 oppure RDWRWTi e EXi con EX(i+1)EN = 00, si può eseguire il servizio degli interrupt. Si servono prima gli NMI, per cui se NMIFF è attivato si passa al gruppo che gestisce la loro routine di servizio.  
Nel primo stato, NMISR, viene avviato un ciclo di fetch, di cui viene ignorata l’istruzione recuperata, e viene anche resettato il flag NMIFF. Nel secondo stato, NMIFETWT, si attende l’inizio della fase di refresh e nel successivo, NMIWT1, si incrementa R attendendo la fine del fetch. L’ultimo stato, NMIWT2, serve per settare il flag NMIIF e attendere il ciclo aggiuntivo al fetch, come di vede in Figura X^4. Alla fine di questa fase, si ritorna in DEC e il DECODER guida l’esecuzione di RST 0066H.

Figura X^4 – Non-Maskable Interrupt Request Operation [Z80.pdf p.24]

Nel caso in cui non siano avvenuti NMI, si controllano gli INT. Se ne sono avvenuti e l’istruzione appena terminata non è EI, IR = 00FBH, o DI, IR = 00F3H, si va nel ciclo INT ACK. Questo avvia il ciclo di interrupt acknowledge nello stato INTSR che resetta anche il flag INTFF per poi passare allo stato INTFETWT che attende la fine del ciclo avviato e poi passa a DEC per l’esecuzione corrispondente.

Figura X^5 – Reset cycle [Z80.pdf p.26]

Per gestire il reset, la FSM Main sfrutta un contatore. Il contatore serve perché la CPU deve accettare il comando di reset ed entrare in uno stato opportuno solo se il segnale nRESET rimane attivo per almeno tre cicli consecutivi di CLK. Il contatore conta questi cicli e al raggiungimento del terzo attiva un segnale di trigger, RST\_TRIG. Alla sua attivazione, la FSM Main entra nel gruppo RESET CYCLE, a prescindere dallo stato corrente, in cui vengono disattivati tutte le uscite e attivati i segnali A\_HZ e DOUT\_HZ.  
La FSM entra nello stato WTRSET in cui si attende che il segnale nRESET si disattivi. Alla sua disattivazione si entra nello stato RSET in cui la CPU attende un ciclo di CLK prima di ripartire con la fase di fetch. Durante questo stato vengono attivati i segnali RSTuFSM, che resetta tutte le sottomacchine rendendole pronte a eseguire i loro cicli, e RSTREG, che resetta tutti i registri.

Figura Y^5 – Diagramma di stato del ciclo RESET CYCLE.