Per rappresentare i diagrammi si 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, i salti a stati o gruppi vengono indicati con una freccia verso il nome della destinazione generalmente azzurra;  
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 principale dello Z80X è gestito da due FSM: Master e Principale.

Master

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

Master gestisce il comportamento del microprocessore quando si verifica una richiesta per 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.  
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 Principale, forza allo stato disattivato i segnali nM1 e nRFSH e attiva tutti i segnali che mettono di bus in alta impendenza. Contemporaneamente segnala alla perifirica richiedente che la richeista 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 Principale.  
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.

Principale

Figura XX – Diagramma di stato della FSM Principale con quadrettati i principali ciclo che vengono espansi nelle immagini successive.##NELLA PAGINA SUCCESSIVA#

La FSM Principale 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.

Tutte le sottomacchine hanno un comportamento comune. Le sottomacchine presentano uno stato di attesa IDLE in cui se il segnale di TRIG è attivo e il CLK a 1 iniziano il loro ciclo rimanendo insensibili ad altri stimoli sino alla fine del ciclo. Si possono far andare le sottomacchine in continuazione tenendo TRIG attivo così la sottomacchina non entra mai nello stato IDLE ma dall’ultimo stato del ciclo passa direttamente al primo ricominciando il ciclo. Per cui quando la FSM Principale attiva il segnale TRIG e la macchina è in attesa inizia il ciclo collegato.  
La sottomacchina segnale la fine del proprio ciclo con il segnale DONE che rimane attivo anche durante l’attesa. In questo modo la FSM Principale può sfruttare due eventi per la temporizzazione: l’inizio del ciclo richiamato quando DONE si disattiva e la fine del ciclo quando si attiva. Per cui Principale mantiene il segnale di TRIG attivo fintantochè non vede disattivarsi DONE per avere la certezza di aver avviato la sottomacchina.  
Il segnale di DONE viene generato quando la macchina si trova nello stato di attesa o negli ultimi stati del ciclo per permettere una corretta lettura del segnale da parte di Principale che essendo stata implementata come una macchina di Moore presenta mezzo ciclo di clock di ritardo sulle variazioni dei sui segnali.  
La macchina che gestisce il fetch dell’opcode ha un segnale aggiuntivo, RFS\_RUN, verso la FSM Principale che avverte quando sta avvenendo la fase di refresh. Questo è utile per attuare la sovrapposizione tra le fasi di fetch e decode.

Presenta un ciclo principale che può essere riassunto come 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 raggruppate in due gruppi. Entrambi in gruppi presentano la stessa struttura: prima uno stato, FET, che avvia la sottomacchina 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 il suffisso EXT che ne segnala la differenza.  
Negli stati di DEC e DECEXT, che eseguono la fase di decode, si attende che la macchina per il fetch concluda completamente il suo ciclo. Nel frattempo viene abilitato il DECODER che leggendo l’IR genera il gruppo FLOWCTRL. Fanno parte di questo gruppo cinque segnali chiamati EXnEN, abbrev. di Execution n Enable, in cui la n è sostituita dall’indice della fase di esecuzione corrispondente, assieme ai segnali IOnMEM, RDnWR, DONE\_EX.

La fase di esecuzione è formata da 5 blocchi elementari che identici a meno dell’indice che ne deternimano l’ordine. Sono cinque poiché al massimo lo Z80, e lo Z80X di conseguenza, eseque al massimo cinque operazioni R/W e 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. Il primo è formato da due stati RDWRn e RDWRWTn, con n 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 sugli FFs degli interrupt e su EX(n+1)EN così da indirizzare lo stato verso il proseguimento corretto dettato da DECODER.  
Lo stato di esecuzione EXn, con n sempre indice della fase, è scandito da un contatore che si incrementa ad ogni evento di CLK. Questo permette di scandire le azioni all’interno dello stato e il DECODER attiva il segnale DONE\_EX al raggiungimento di valori specifici che corrispondono all’effettiva durata collegata all’istruzione che si sta svolgendo e all’uscita dallo stato EXn.  
La selezione di un percorso od un altro è determinata dal valore di EXnEN 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.  
Può avvenire che la un’istruzione abbia EX1EN = 00, il che significa che è un’operazione istantanea svolta direttamente nella fase di decode. Un esempio sono le istruzioni di caricamento di un registro un altro come LD A, B oppure le operazioni aritmetiche a 8 bit come ADD A, B.  
Durante l’ultima fase di esecuzione, EXEC5, non viene letto nessun

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 all’esterno. Lo Z80X può uscire da questo stato solo per mezzo di un NMI o di un INT se abilitato oppure con RESET. 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 mette in stand-by la CPU mantenendo comunque la RAM aggiornata.  
Si può inoltre fare mettere la CPU in uno stato di power-down così da farle consumare meno corrente se alla fine di un ciclo di HALT si toglie 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.

L’esecuzione di un INT e si sia nel 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 RDWRWTn e EXn con EX(n+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 resettato il flag NMIFF. Nel secondo stato, NMIFETWT, si attende l’inizio della fase di refresh e nel successivo, NMIWT1, si incrementa R e si attende 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 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 Principale 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 di CLK consecutivi. Il contatore conta questi cicli e al raggiungimento del terzo attiva un segnale di trigger RST\_TRIG. Alla sua attivazione, la FSM Principale 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 e vengono attivati i segnali RSTuFSM, che resetta tutte le sottomacchine rendendole pronte a eseguire i solo cicli, e RSTREG, che resetta tutti i registri.

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