Tutte le uFSM hanno lo scopo di eseguire un ciclo preciso svolto dalla CPU e di aggiornare dei valori internamente a Z80X. I cicli in esame sono gli unici che durante il normale funzionamento dello Z80X coinvolgono l’esterno dell’entity. Per cui nella descrizione che ne segue terrò conto del loro port, del ciclo che devono svolgere e dell’FSM che lo implementa e del loro compito all’interno dello Z80X.

Tutte le sottomacchine hanno uno schema e un comportamento comune.

Le uFSM presentano tutte la coppia di segnali TRIG e DONE, seguite dal suffisso della macchina, per il controllo all’interno della struttura annidata delle FSMs. Si aggiunge il segnale STOP che se attivato impedisce alla macchina di passare allo stato successivo.

Le sottomacchine presentano uno stato di attesa IDLE in cui se il segnale di TRIG è attivo e il CLK è a 1 iniziano il loro ciclo rimanendo insensibili ad altri stimoli su TRIG sino alla fine del ciclo. Per cui quando la macchina è in attesa e la FSM Main attiva il segnale TRIG inizia il ciclo collegato.  
Si possono far andare le uFSM 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.  
La uFSM segnala la fine del proprio ciclo con il segnale DONE che rimane attivo anche durante l’attesa. In questo modo la FSM Main 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 Main mantiene il segnale di TRIG attivo fintantoché non vede disattivarsi DONE per avere la certezza di aver avviato la sottomacchina correttamente.  
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 Main. Il motivo è che siccome Main è stata implementata come una macchina di Moore presenta mezzo ciclo di CLK di ritardo sulle variazioni dei sui segnali.

Instruction Opcode Fetch, OPFET

Figura Y – Schema dell’entity OPFET

La macchina svolge il ciclo di fetch, che dura di base 4 T-cycles, in cui la CPU legge dalla memoria l’opcode all’indirizzo puntato da PC e lo carica in IR. Dopodiché esegue il refresh della RAM mettendo sul bus l’indirizzo di refresh, contenuto in R, e attivando gli opportuni segnali.

Con il segnale A\_LD carica in A LATCH il PC o R selezionandoli con il segnale PCnRFSH. Mentre con il segnale DIN\_LD carica in IR il valore di DIN, per cui è collegato al segnale SHIFT del registro.

Per permettere la sovrapposizione delle fasi di fetch e decode, la macchina fornisce alla FSM Main un ulteriore segnale, RFS\_RUN, che si attiva nello stato precedente l’inizio della fase di refresh e rimane attivo fino alla fine della fase. L’attivarsi di questo segnale indica alla macchina di passare dallo stato di attesa a quello di decodifica: da FETWT a DEC o da FETEXTWT a DECEXT.

La macchina oltre al comportamento descritto dal diagramma in Figura X, resetta IR nello stato precedente il caricamento cioè in CHKWT, che è anche lo stato in cui campiona nWAIT per entrare o meno nel ciclo di attesa. Il RSTIR viene mascherato con il segnale FIRST\_FET che viene attivato solo in FET.  
La macchina carica invece il valore letto in IR durante RD.

Figura X – Instruction Opcode Fetch [Z80.pdf p.20]

Figura Y^2 – Digramma di stato della uFSM OPFET.

Memory Read or Write Cycles, MEMRDWR

Figura Y^3 – Schema dell’entity MEMRDWR

La macchina svolge il ciclo di lettura o scrittura su memoria, che dura di base 3 T-cycles.  
In questo ciclo la CPU legge dalla memoria un dato all’indirizzo puntato dal valore sul bus interno DATA16 e lo carica in MDR. Oppure scrive all’indirizzo presenta sul bus DATA16 il valore presente sul bus interno DATA.

Il verso dell’operazione è determinato dal segnale RDnWRFF, come si vede dal suffisso -FF è la versione campionata da un FF del segnale d’ingresso RDnWR. Il FF campiona il valore dell’ingresso mentre il segnale ENFF è attivo, cioè durante IDLE, WRA o RDWR.  
Il ciclo della macchina rimane pressoché invariato in base al valore di RDnWRFF a meno della coppia di stati INITRD e WRD. Il primo avvisa la memoria che sta avvenendo una lettura e il secondo carica il valore di DATA sul DOUT LATCH. Per il resto cambia solamente che nRD sia attivato, nel caso in cui RDnWRFF = 1, o alternativamente che nWR sia attivato, con RDnWRFF = 0. Nel diagramma di Figura Y^4 questo è mostrato per mezzo del segnale fittizio RDWR.

La scrittura del valore sul bus D avviene appunto nello stato WRD mentre quella dell’indirizzo nello stato WRA con l’attivazione di A\_LD. La lettura e il caricamento di MDR avvengono nello stato RDWR se RDnWRFF = 1.

Figura X^2 – Memory Read or Write Cycles [Z80.pdf p.25]

Figura Y^4 – Digramma di stato della uFSM MEMRDWR.

Input or Output Cycles, IORDWR

Figura Y^5 – Schema dell’entity IORDWR

La macchina svolge il ciclo di lettura o scrittura sulle periferiche, che dura di base 4 T-cycles.  
La macchina è la controparte per gli I/O di MEMRDWR per cui ha lo stesso funzionamento a meno di avere un ciclo di attesa in più, WA, in cui non avviene nessun cambiamento delle uscite.  
Inoltre controlla nIORQ invece di nMREQ e attiva i segnali nRD o nWR avviene nello stesso stato e non sfalsati. Per questo il ciclo della macchina è esattamente lo stesso indifferentemente dal valore di RDnWRFF.

Di conseguenza, la CPU legge un dato all’indirizzo puntato dal valore sul bus interno DATA16 e lo carica in MDR. Oppure scrive all’indirizzo presente sul bus DATA16 il valore presente sul bus interno DATA.  
Il verso dell’operazione è determinato dal segnale RDnWRFF. Il FF campiona il valore dell’ingresso mentre il segnale ENFF è attivo, cioè durante IDLE, WRA o RDWR. Come per MEMRDWR, l’attivazione di nRD o nWR in base a RDnWRFF è mostrata per mezzo del segnale fittizio RDWR nella Figura Y^6.

La scrittura del valore sul bus D avviene appunto nello stato WRD se RDnWRFF = 1 mentre quella dell’indirizzo nello stato WRA con l’attivazione di A\_LD. La lettura e il caricamento di MDR avvengono nello stato RDWR se RDnWRFF = 1.

Figura X^3 – Input or Output Cycles [Z80.pdf p.25]

Figura Y^6 – Digramma di stato della uFSM IORDWR.

Interrupt Request/Acknowledge Cycle, INTRQ

Figura Y^6 – Schema dell’entity INTRQ

La macchina svolge il ciclo di interrupt acknowledge in caso sia avvenuto un INT, che dura di base 5 T-cycles.  
La CPU esegue un ciclo simile a quello di fetch in cui carica il valore di PC anche se non è utile. Inoltre invia il segnale di interrupt acknowledge attraverso l’attivazione della combinazione di segnali nIORQ e nM1. Dopodiché legge dal bus il valore dell’istruzione da svolgere se in Mode 0 o la parte bassa dell’indirizzo se in Mode 2. In ogni caso il valore letto viene caricato in IR e poi sarà il DECODER a utilizzare sapientemente IR in base alla modalità impostata.

A differenza del ciclo di fetch, presenta due stati di attesa in più, TWA1 e TWA2. In realtà la macchina attende davvero in T2 e TWA1 mentre in TWA2 attiva il segnale nIORQ.

La scrittura dell’indirizzo avviene nello stato WRA con l’attivazione di A\_LD. La lettura e il caricamento di MDR avvengono nello stato RD.

Figura X^4 – Interrupt Request/Acknowledge Cycle [Z80.pdf p.23]

Figura Y^8 – Digramma di stato della uFSM INTREQ.