Figura X – Schema dell’entity SS\_DRIVER

Lo scopo dell’uso del display a 7 segmenti è quello di far comunicare lo Z80X con l’utente per cui deve permettere un utilizzo molto flessibile. A questo scopo è stato sviluppato il driver per il display.

Il driver chiamato SS\_DRIVER presenta una struttura a livelli in base alle azioni da compiere.

Nel livello più basso si trova un decoder da ASCII ai segnali di controllo dei segmenti. Questo decoder chiamato SSegsASCII riceve in ingresso il codice ASCII a 7 bit del carattere da visualizzare e ritorna il segnale per il 7 segmenti a 8 bit in cui il più significativo rappresenta il punto decimale. La presenza di questo driver permette di non preoccuparsi su come mostrare una cifra, una lettera o un simbolo sul display ma basta solamente fornire il codice ASCII.

Al livello superiore vi è l’entity ASCIISSegsDriver che permette il controllo di tutte le cifre. Presenta due generic: DIGIT, che è il numero di cifre del display così da rendere universale il controllore; CLK\_HZ, che è la frequenza in hertz del clock per generare correttamente i segnali di temporizzazione. L’entity si occupa di gestire le cifre per mezzo di una logica multiplexata con gli anodi.  
Per cui genera un vettore con un bit per ogni cifra, in cui è presenta al massimo un solo segnale di attivazione con cui attiva a turno ogni cifra. In questo modo mantenendo un tempo di circa 1 ms per ogni cifra si sfrutta la persistenza dell’occhio così far sembrare accese tutte le cifre contemporaneamente. Ma allo steso tempo risparmiando sia bus per i segmenti che corrente. Inoltre l’entity fornisce anche il valore decodificato della cifra corrispondente attraverso SSegsASCII. Utilizzando poi il bit più significativo della cifra in ingresso, che rimarrebbe inutilizzato, come bit per l’attivazione del punto decimale permette di non sprecare una cifra come punto. I caratteri gli vengono passati ordinati per mezzo di un unico vettore di tanti byte quante sono le cifre.  
Inoltre l’entity mette a disposizione un vettore per abilitare o meno le cifre così da spegnerne selettivamente una o più.

Sopra quest’entity vi è l’entity CircBuffSSegs che implementa un buffer circolare per il display. Basandosi su una RAM che immagazzina i dati, li legge uno per volta e li inserisce in uno shift register SIPO grande tanti byte quante sono le cifre. Così in caso di messaggi lunghi, i caratteri si possono fare scorrere sul display.  
La RAM è indirizzata da 0 sino al valore dell’ingresso LIM che segna l’indirizzo dell’ultimo carattere da visualizzare.  
L’entity fornisce in uscita il segnale LAST che segnala al livello superiore quando si sta leggendo l’ultimo carattere e cioè quando l’indirizzo puntato è pari a LIM.  
Lo shift dei caratteri è gestito dall’ingresso SH e quindi dall’entity del livello superiore.

L’ultimo livello è occupato da SS\_DRIVER che gestisce quattro diversi comportamenti del display: la dimensione del messaggio e lo scorrimento, la velocità di scorrimento, il lampeggio selettivo delle cifre e la funzione di allineamento, detta adjust.  
L’entity si aspetta che il messaggio sia caricato autonomamente sulla RAM dall’utilizzatore e che questo comunichi la dimensione del messaggio inserendo il numero di caratteri sul vettore NCHAR. L’entity abiliterà lo scorrimento dei caratteri se questi superano il numero di cifre del display. La frequenza dello scorrimento è selezionata dall’esterno per mezzo del vettore SHDIV come divisione di una frequenza fondamentale e massima. La frequenza massima è settata per mezzo del generic SH\_MAX\_FREQ\_HZ assieme al numero di bit del divisore N\_SH. L’utilizzatore può anche forzare lo scorrimento per mezzo dell’ingresso FSH, abbrev. di Force Shift. Il segnale di fine messaggio dell’entity sottostante è riportato all’esterno.  
L’entity gestisce anche il lampeggio delle cifre selezionate dal vettore ENSTROBE tutte con la stessa frequenza impostata per mezzo di STROBEDIV come divisione di una frequenza massima. La frequenza e il numero di bit del divisore sono settati con i generic STROBE\_MAX\_FREQ\_HZ e N\_STROBE rispettivamente.  
In aggiunta l’entity permette di aggiustare a sinistra l’inizio del messaggio sul display e di spegnere l’intero display con il segnale ENOUT. Queste operazioni sono utili quando si scrive un nuovo messaggio sullo schermo. Per cui prima si spegne la visualizzazione, si scrive il nuovo messaggio in RAM, si precarica il buffer circolare con i primi caratteri e poi si riattiva il display.  
Per attuare l’adjust, l’entity fa uso di una piccola FSM. La macchina rimane nello stato IDLE fintantoché l’ingresso ADJ non è attivato. Quando ciò si verifica, entra nello stato ADJR in cui resetta il buffer circolare ed un contatore di modulo pari al numero di cifre. Poi nello stato ADJS abilita il contatore, che mantiene attivo il segnale di scorrimento del buffer circolare sino al completamento del conteggio. A questo punto la macchina passa allo stato ADJW in cui attende che il segnale ADJ si disattivi e nel frattempo attiva il segnale ADJDN, abbrev. di Adjust Done, disattivando lo scorrimento del buffer. Dopodiché torna in IDLE.