*Esercizio 6 – Sistema di testing*

*Esercizio 6.1*

Progettare, implementare in VHDL e verificare mediante simulazione un sistema in grado di testare in maniera automatica una macchina combinatoria M avente 4 ingressi e 3 uscite binarie sottoponendole N ingressi diversi (si considerino una macchina M e un numero di input N a scelta dello studente).

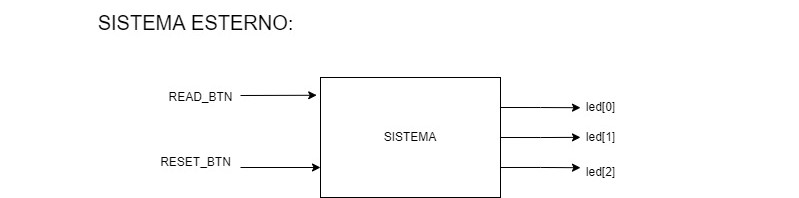
Gli N valori di input per il test devono essere letti da una ROM, in cui essi sono precaricati, in corrispondenza di un segnale *read*. Le N uscite fornite della macchina in corrispondenza di ciascuno degli input devono essere memorizzati in una memoria interna, che deve poter essere svuotata in qualsiasi momento in presenza di un segnale di *reset*.

*Esercizio 6.2*

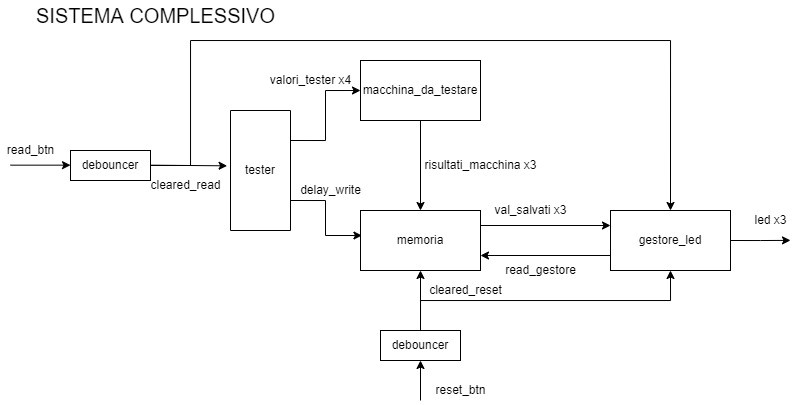
Sintetizzare ed implementare su board il componente sviluppato al punto precedente, utilizzando due bottoni per i segnali di *read* e *reset* rispettivamente e i led per la visualizzazione delle uscite della macchina istante per istante.

Scelte progettuali:

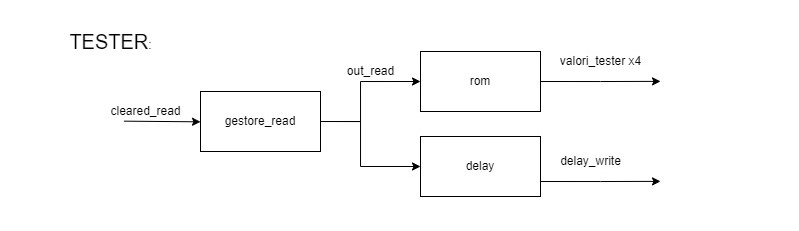
L’esercizio richiedeva di progettare un sistema di testing per una macchina combinatoria, a partire da dei valori di input per la macchina precaricati in una ROM, si visualizza l’output della macchina combinatoria a tali valori tramite dei LED. Gli output vengono visualizzati a intervalli di 1s. Per poter iniziare il testing della macchina occorre un segnale di RESET, necessario per svuotare la memoria, ed un segnale di READ, per presentare in ingresso alla macchina gli input presenti nella ROM.



Il sistema è stato progettato mediante un approccio strutturale:



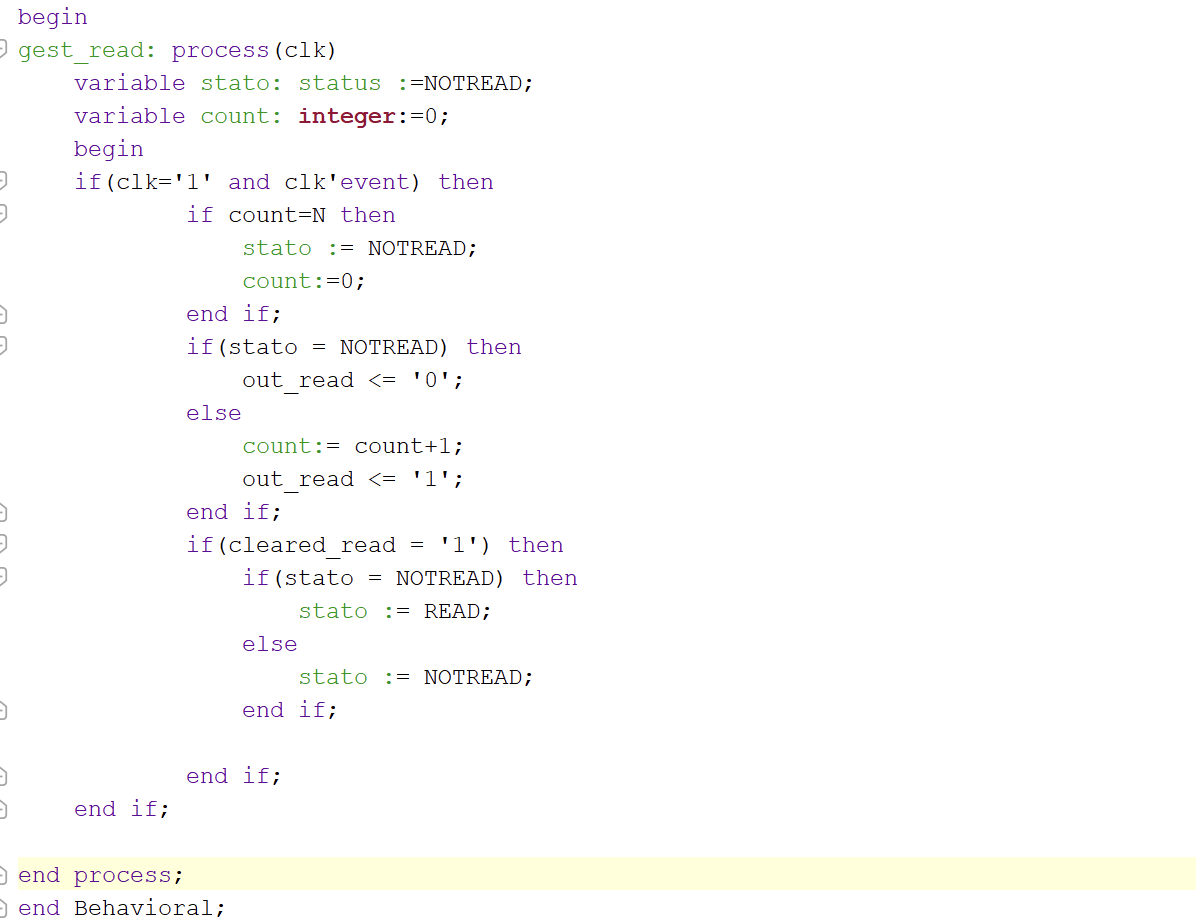
Tester:



Il tester è anch’esso realizzato con approccio strutturale. Lo scopo di questo componente è quello di presentare in uscita dei valori da dare in input alla macchina che si vuole testare ed inviare un segnale di scrittura per il salvataggio dei risultati.Il sistema è composto da:

-Gestore read:



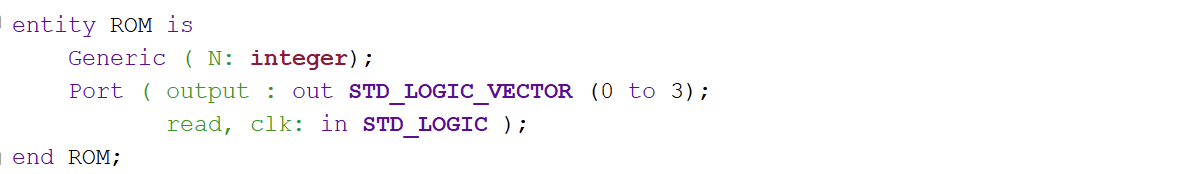


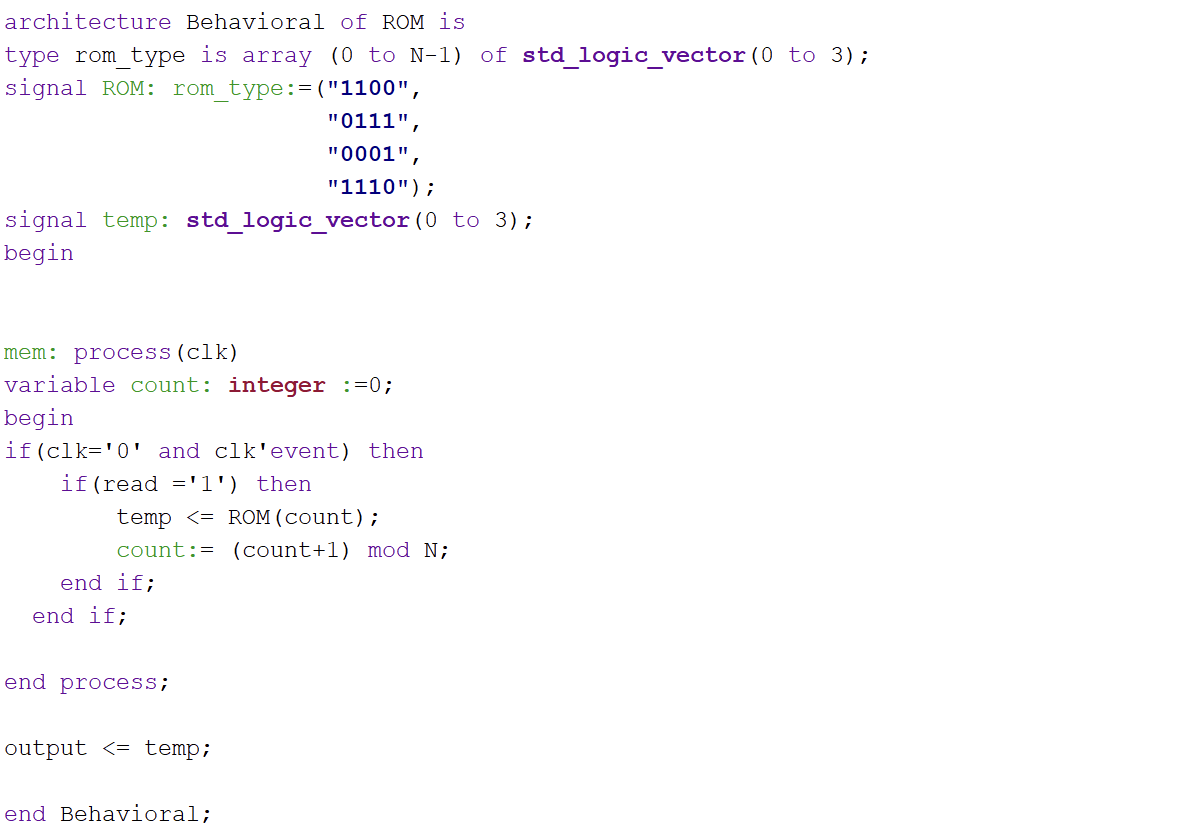
A partire dal segnale ripulito di READ viene prodotto un segnale di OUT\_READ che rimane alto per tanti periodi di clk quanti sono gli input di test: N\*Tclk. E’ realizzato come una macchina a stati:

-stato NOTREAD: in tale state il valore di out\_read è pari a 0 e rimane in attesa di un segnale cleared\_read, appena questo arriva si passa nello stato di READ.

-stato READ: a tale stato è associata l’uscita out\_Read pari a 1; il sistema rimane in questo stato per N colpi di clock per poi tornare allo stato NOTREAD.

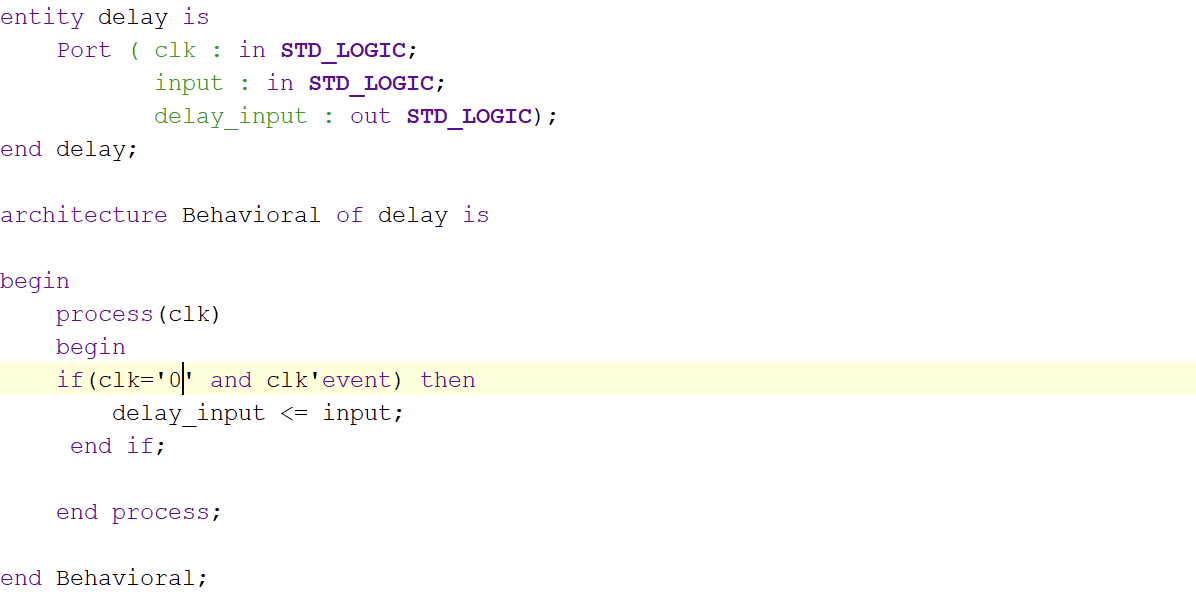
-ROM:





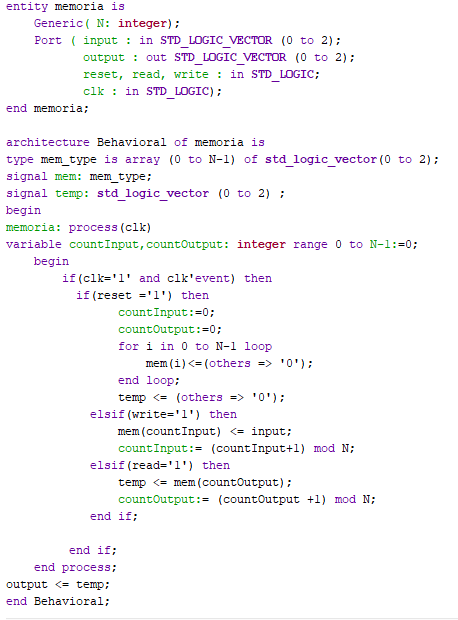
E’ stata realizzata come una memoria precaricata, che all’arrivo del segnale di read visualizza in uscita il valore puntato ed incrementa il puntatore ai valori memorizzati. Il generic permette di memorizzare i valori al variare di N.

-DELAY: Il blocco di delay è necessario per poter ritardare il segnale di write in memoria, in questo modo si sincronizza l’uscita della macchina combinatoria al segnale di scrittura. Si suppone che la macchina combinatoria abbia un ritardo inferiore al Tclock/2; se così non fosse allora dovremmo aumentare il periodo del clock con un divisore in frequenza.

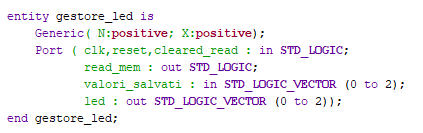


Il blocco di delay e la ROM agiscono sul fronte di discesa del clock in modo che il ritardo ottenuto sia tale da campionare il valore di uscita della macchina quando è stabile e non sulla transizione. Il segnale di write, infatti, varia sul fronte di discesa del clock, mentre la scrittura in memoria avviene sul fronte di salita.

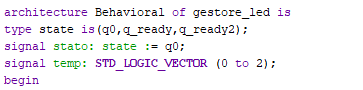
memoria: è realizzata in maniera comportamentale. La scrittura avviene con la delay\_write (che dura N\*Tclk), quindi permette di scrivere N valori in uscita dalla macchina combinatoria. La lettura avviene con la read\_gestore (che dura Tclk), la quale permette di scorrere i valori in memoria con periodo di 1s. Agiscono due puntatori: uno per la lettura ed uno per la scrittura.

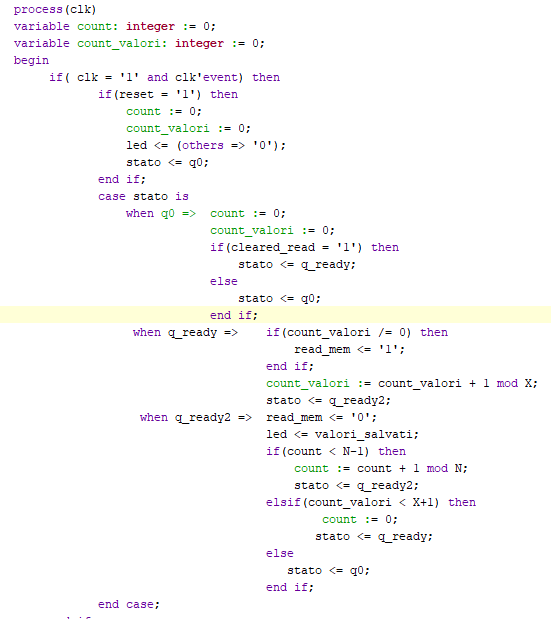


Gestore\_led:



È realizzato come una macchina a stati che gestisce la visualizzazione degli output, visualizzandoli ad intervalli di 1s. Il generic N permette di gestire, tramite una variabile count, l’intervallo di visualizzazione degli output. Il generic X permette di stabilire, tramite la variabile count\_valori, il numero di valori da visualizzare.



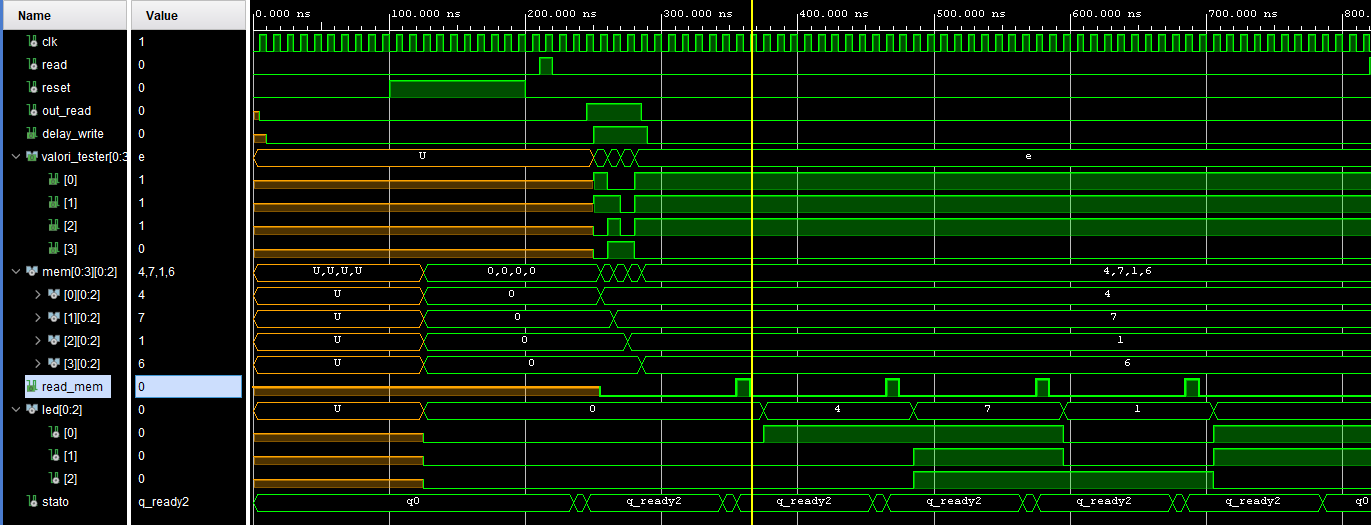


Stato q0: è lo stato in cui si arriva dopo ogni segnale di RESET o dopo la visualizzazione completa dei valori in memoria. Vengono resettati i valori di count, count\_valori e si rimane in attesa di un segnale di READ per poter passare allo stato q\_Ready.

Stato q\_Ready: si confronta il valore di count\_valori con 0 per poter stabilire se è la prima iterazione, in questo caso si aspetta che la macchina elabora i dati in ingresso per poi passare alla visualizzazione effettiva, altrimenti si alza il segnale di lettura della memoria. Si incrementa, poi, il valore di count\_valori e si passa allo stato q\_Ready2.

Stato q\_Ready2: si abbassa il segnale di lettura della memoria e si mostrano i risultati, attendendo un numero di impulsi di clock pari ad N tramite la variabile count, prima di procedere alla visualizzazione del risultato successivo. Si effettua, poi, un controllo per stabilire se il valore visualizzato è l’ultimo o meno tramite count\_valori: se è così, allora si passa allo stato di q0; altrimenti si passa in q\_Ready.

Simulazione:



Sintesi su FPGA:

Per la sintesi su fpga sono stati inseriti i debouncer per i pulsanti di READ e RESET.

