PROVA FINALE

PROGETTO DI RETI LOGICHE

Studente:	Lu Valeria
Codice studente:	
Matricola:	
Anno accademico:	2023-2024

INDICE

1. INTRODUZIONE

L'obiettivo finale di questo progetto è implementazione di un modulo hardware, descritto in VHDL, che si rispetti il funzionamento riportato seguente.

1.1 Funzionamento generale

Supponiamo di avere in memoria una sequenza di parole, per semplicità lo rappresentiamo con i valori decimali:

20	0	0	0	100	0	5	0	0	0	0	0

La sequenza di parole da elaborare è memorizzata ogni due byte:

20	0	0	0	100	0	5	0	0	0	0	0

Le regole da seguire sono queste:

- ogni parola ha un valore compreso tra 0 e 255
- il valore 0 all'interno della sequenza indica "il valore non è specificato"
- completare la sequenza

  se il valore letto è zero sostituirlo con l'ultimo valore letto diverso da zero, appartenente alla sequenza
- inserendo nel byte subito successivo

  un valore di "credibilità" per ogni valore della sequenza

- il valore di credibilità è sempre maggiore o uguale a 0, viene inizializzato a 31 ogni volta che il valore letto della sequenza è non zero, mentre viene decrementato di 1 ogni volta che si incontra una parola di valore 0.
- se il primo dato della sequenza è pari a zero, il suo valore rimane tale e il valore di credibilità deve essere posto a 0 (zero). Lo stesso succede fino al raggiungimento del primo dato della sequenza con valore diverso da zero.

Seguendo le regole sopra descritte, la sequenza finale sarà:

20	31	20	30	100	31	5	31	5	30	5	29

1.2 L'Interfaccia del componente

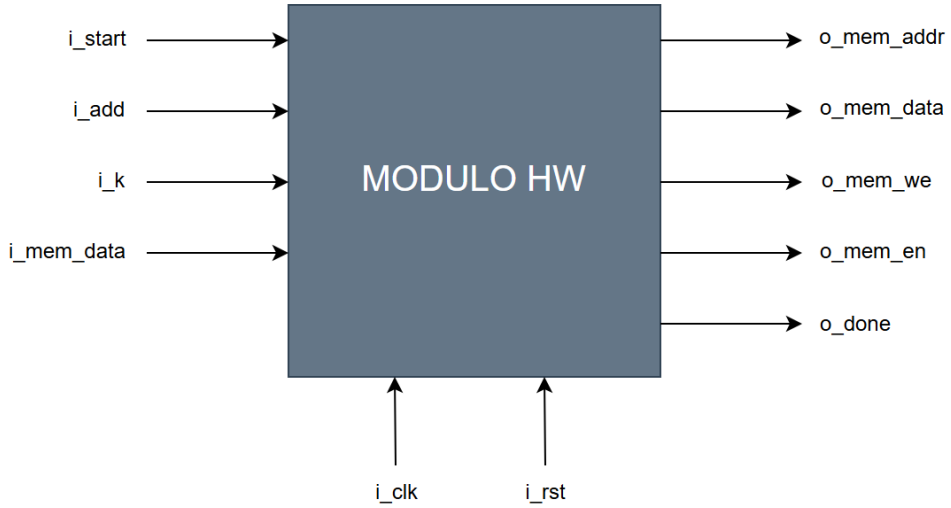
Il modulo da implementare ha 4 ingressi primari:

i_start	1 bit	segnale di START
i_add	16 bit	segnale che rappresenta l'indirizzo dal quale parte la sequenza da elaborare
i_k	10 bit	segnale che rappresenta lunghezza della sequenza
i_mem_data	8 bit	segnale che arriva dalla memoria e contiene il dato letto

E ha 5 uscite primari:

o_mem_addr	16 bit	segnale che manda l'indirizzo alla memoria
o_mem_data	8 bit	segnale che contiene il dato che verrà successivamente scritto
o_mem_en	1 bit	segnale di ENABLE da dover mandare in memoria per poter communicare, sia in lettura che scrittura
o_mem_we	1 bit	segnale di WRITE ENABLE da dover mandare in memoria per poter scriverci
o_done	1 bit	segnale di DONE che communica fine dell'elaborazione

Inoltre, il modulo ha un segnale di CLOCK (i_clk) e un segnale di RESET (i_rst), entrambi unici per tutto il sistema.



1.3 Dettagli implementativi

Il modulo interfaccia con una memoria e legge un messaggio costruito da una sequenza di K parole W, quest'ultimo è memorizzato a partire da un indirizzo specificato ADD, ogni due byte, quindi ADD, ADD+2, ADD+4, ..., ADD+2*(K-1).
Il valore di credibilità associato ad ogni parola viene memorizzato in memoria nel byte subito successivo, quindi ADD+1, ADD+3, ...

All'istante iniziale, quello relativo al reset del sistema, l'uscita DONE deve essere 0.

Una volta che RESET torna a zero, il modulo partirà nella elaborazione quando un segnale START in ingresso verrà portato a 1. Il segnale di START rimarrà alto fino a che il segnale di DONE non verrà portato alto; al termine della computazione (e una volta scritto il risultato in memoria), il modulo alzerà (portare a 1) il segnale DONE che notifica la fine dell'elaborazione. Il segnale DONE rimane alto fino a che il segnale di START non è riportato a 0. Un nuovo segnale START non può essere dato fin tanto che DONE non è stato riportato a zero.

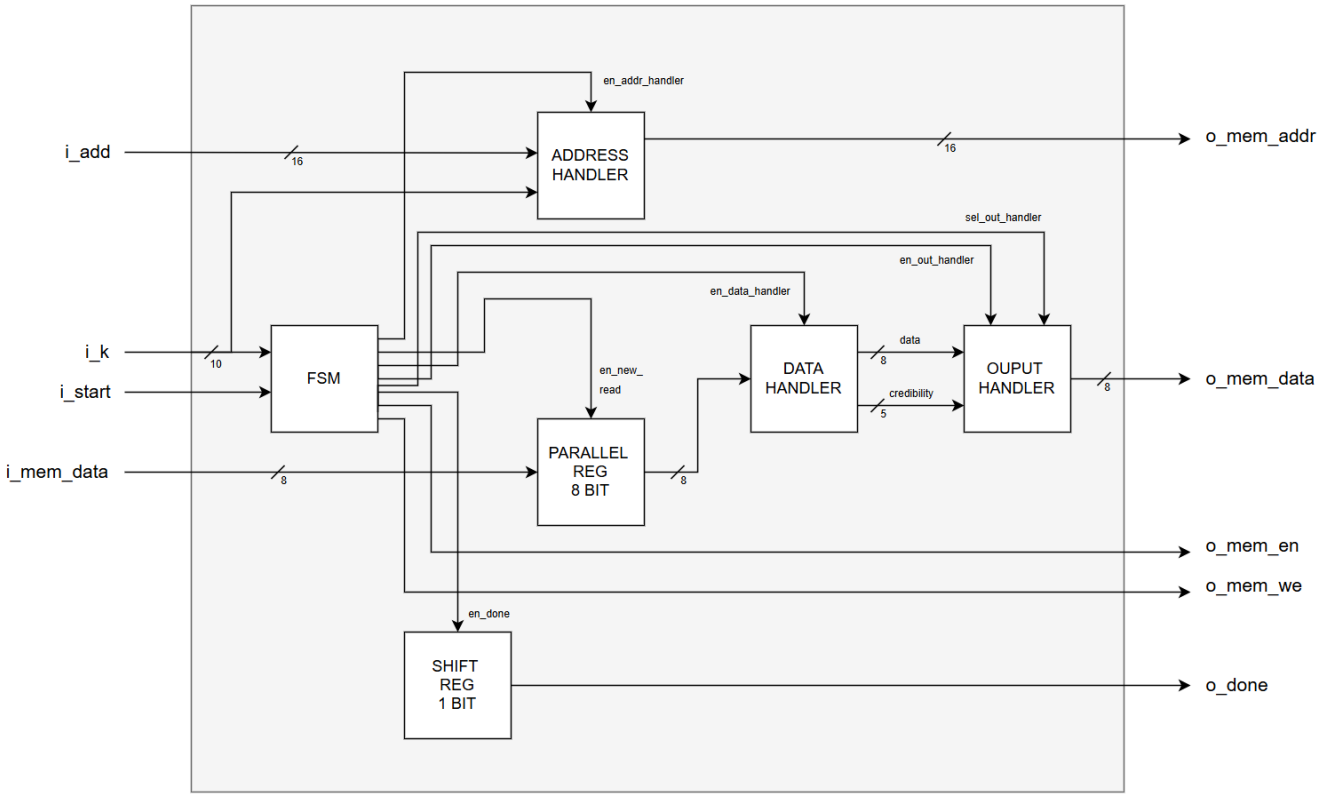
Il modulo è stato progettato considerando che prima del primo START=1 verrà sempre dato il RESET (RESET=1).
Una seconda (o successiva) elaborazione con START=1 non dovrà attendere il reset del modulo. Ogni qual volta viene dato il segnale di RESET (RESET=1), il modulo viene re-inizializzato.

Quando il segnale di START viene posto ad 1 (e per tutto il periodo in cui esso rimane alto) sugli ingressi ADD e K sono posti il primo indirizzo e la dimensione della sequenza da elaborare. Il modulo prima di alzare il segnale di DONE deve aggiornare la sequenza ed i relativi valori di credibilità al valore opportuno seguendo la descrizione generale del modulo.

2. ARCHITTETURA

Il componente è stato progettato e simulato utilizzando Xilinx Vivado come strumento, l'FPGA target è la Xilinx Artix-7 FPGA xc7a200tfbg484-1, la strategia di implementazione adottata dalla progettista è quella modulare.

2.1 Moduli



NOTA: lo schema di moduli riportato sopra è uno schema opportunamente semplificato per una visualizzazione più pulita. Il segnale di clock e il segnale di reset sono unici per sistema, e sono collegati ai vari moduli (tutti ad eccetto il modulo output handler che è puramente combinatorio).

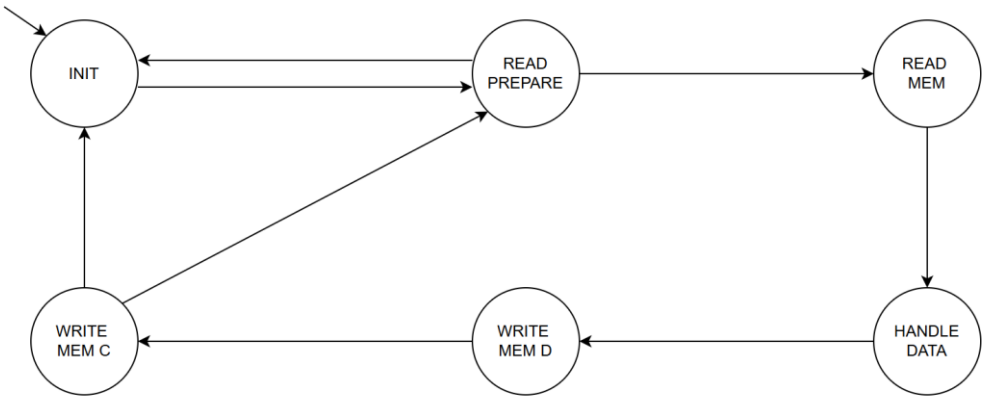
Il componente è composto da 4 sotto moduli principali:

a. FSM: è la macchina a stati che “controlla” i vari moduli e registri, il cui schema in termini di diagramma degli stati, ha 6 stati (per dettagli si veda il paragrafo 2.2).
b. ADDRESS HANDLER
   modulo che ha il compito di elaborare il segnale di uscita che manda l’indirizzo alla memoria e contiene a suo in interno un contatore da 11 bit per poter tenere traccia dell’indirizzo di memoria correntemente in lettura o in scrittura, tale valore viene usato poi in altri moduli per verificare la fine dell’elaborazione, settato a 0 in caso di reset e del fine di elaborazione.
   L’indirizzo di memoria in uscita viene calcolata sommando il valore memorizzato dal contatore con l’indirizzo di memoria di partenza (i_add).
c. DATA HANDLER
   modulo che computa il valore del dato e il valore della credibilità, rispettando le regole descritte nel paragrafo 1.1.
   Memorizza al suo interno come supporto di computazione un valore dell’ultimo dato letto valido, ovvero positivo, e dell’ultimo valore di credibilità, entrambi in caso di reset del sistema o del fine dell’elaborazione vengono posti a zeri.
d. OUTPUT HANDLER
   modulo che contiene un multiplexer a suo interno e produce come output il valore del dato da scrivere in memoria se il segnale di selezione è a 0, il valore di credibilità se il segnale di selezione è a 1.

inoltre, ha 2 registri:

e. PARALLEL REG 8 BIT
   Parallel shift register a 8 bit, viene usato per memorizzare il nuovo dato letto dalla memoria.
f. SHIFT REG 1 BIT
   Shift register a 1 bit, viene usato per memorizzare il valore del done, tale valore viene portato al segnale di uscita o_done e viene utilizzato da altri moduli in caso di necessità.

2.2 Macchina a stati



La macchina a stati descritto nel paragrafo precedente ha complessivamente 6 stati.

Lo stato INITIAL (INIT della figura) è lo stato di reset, rimane in questo stato finché il segnale di start non viene posto a 1, e mantiene a 0 tutti gli segnali di controllo verso vari moduli.

Una volta che segnale di start viene posto a 1, si passa allo stato READ PREPARE che effettua la richiesta della lettura di un dato dello specifico l'indirizzo di memoria computato in caso che il segnale i_k, ovvero lunghezza della sequenza di dati da elaborare, non sia 0; altrimenti, alza il segnale en_done e ritorna allo stato di reset.

Una volta letto il dato si passa allo stato READ MEM che memorizza nel registro a 8 bit il dato appena letto, lo stato successivo HANDLE DATA effettua il computo del valore del dato e della sua credibilità da scrivere in memoria.

Infine, ultimi passi del ciclo si procedono con le rispettive scritture in memoria dei due valori, lo stato WRITE MEM D per il dato e WRITE MEM C per la credibilità, ovviamente computando l'indirizzo di memoria in uscita corrispondente.

Dopo lo stato WRITE MEM C, due possibili situazioni possono accadere: se la macchina ha terminato l'elaborazione, alza il segnale en_done e passa allo stato INITIAL; altrimenti prosegue verso stato READ PREPARE e ripete il ciclo finché non sia concluso l'elaborazione dell'intera sequenza.

3. RISULTATI SPERIMENTALI

Il componente supera una fase di sperimentazione, viene sottoposto sia a simulazioni comportamentale pre-sintesi che post-sintesi (funzionale), con opportuni test bench; risulta quindi correttamente sintetizzabile.

3.1 Sintesi

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	66	0	134600	0.05
LUT as Logic	66	0	134600	0.05
LUT as Memory	0	0	46200	0.00
Slice Registers	44	0	269200	0.02
Register as Flip Flop	44	0	269200	0.02
Register as Latch	0	0	269200	0.00
F7 Muxes	0	0	67300	0.00
F8 Muxes	0	0	33650	0.00

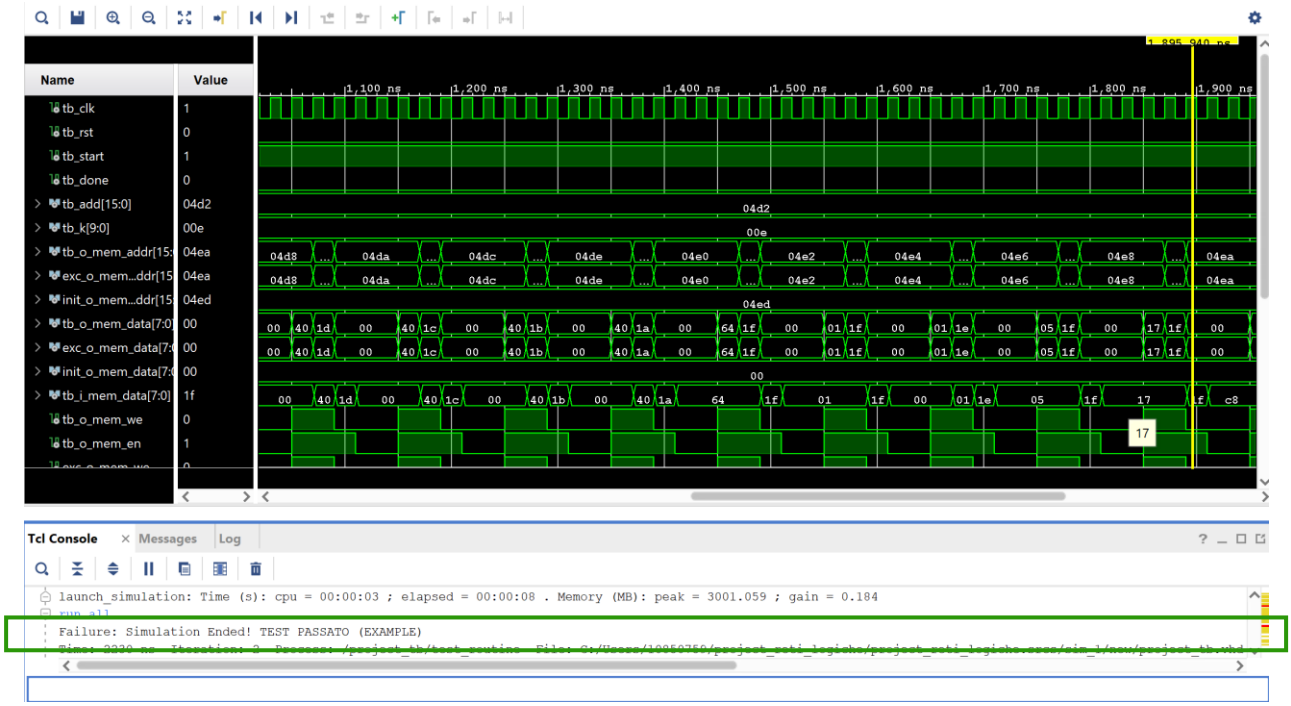
Si osservi che il modulo non generi nessun latch non volontariamente voluto.

3.2 Simulazioni

Tra vari test sottoposti, questo componente ha passato:

- un test bench che ha come l'ingresso una sequenza di 0 parole, il componente riesce a riconoscere il fatto e termina l'elaborazione.
- un test bench che verifica la capacità di elaborare più scenari in ingresso in sequenza, il componente non incontra nessun problema.
- un test bench che effettua un reset durante la lettura dalla memoria.
- un test bench che verifica la capacità di riprendere una nuova esecuzione in seguito al riconoscimento di un segnale i_rst asincrono.
- i test di normali elaborazioni e altri casi.

Il componente risulta funzionante passando tutti i test sopra descritti.



4. CONCLUSIONI

Il componente HW descritto soddisfa tutti i requisiti della specifica, è stato progettato e implementato attentamente seguendo tutte le fasi necessarie.

Il componente può essere ulteriormente ottimizzato, si noti che tale operazione non sarà difficoltoso grazie a una progettazione curata dell'architettura interna, inoltre un ulteriore modifica alla logica di un qualsiasi sotto-modulo risulta indipendente dagli altri sotto-moduli, il che porta ad una larga flessibilità.